Using GANs to create synthetic datasets for fake news detection models

Bruno Gonçalves Vaz
Master's Degree in Data Science
Department of Computer Science
2022

Supervisor
Álvaro Figueira, Auxiliary Professor, Faculty of Sciences

Declaração de Honra

Eu, Bruno Gonçalves Vaz, inscrito(a) no Mestrado em Ciência de Dados (Data Science) da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor.

Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Bruno Gonçalves Vaz

Porto, 03/12/2022

Abstract

Fake news consist of intentionally false information and their proliferation has been ever-increasing in recent years. They are frequently used for political or financial gain, causing severe social problems, such as the adoption of non-scientific alternative medicines, extreme political standpoints, or pyramid schemes. Moreover, the high reachability they now have, powered by social media platforms which allow their widespread and fast dissemination, makes it crucial to find novel ways to tackle this problem.

Machine Learning (ML) approaches have been already proposed, where models are trained with the purpose of detecting fake news. Nonetheless, most of the news datasets in which these models train are extremely imbalanced, having substantially more real news than fake ones. As such, ML models struggle to generalize properly, causing them not to be accurate at detecting fake news.

One promising approach is to artificially balance the ratio between fake news and real news by using synthetic data. When data are scarce or of poor quality, synthetic data can be used, for example, to improve the performance of ML models. Synthetic data are artificially generated from real data and have the same statistical properties as real data. However, unlike real data that is measured or collected in the real world, synthetic data is generated by computer algorithms. There are several methods for generating synthetic data, *e.g.* Random Oversampling (ROS), Synthetic Minority Over-sampling Technique (SMOTE), or Gaussian Mixture Model (GMM). However, in recent years, a new and promising generative model has emerged. Generative Adversarial Networks (GANs) are a state-of-the-art deep generative model that can generate novel synthetic samples that follow the underlying data distribution of the original dataset. They are widely used and provide very good results in image domains, but are still poorly explored in domains with tabular data.

In this dissertation, we conducted a thorough literature review on the topics of GANs (especially for tabular data), methods for generating synthetic data, and quality assessment of synthetic data, which, to our knowledge, have not been explicitly combined in any of the relevant literature. Having laid the groundwork for our work, we demonstrated the potential of synthetic data to improve the performance of ML models in detecting fake news, as well as the best performing GAN architectures. To this end, we conducted an experiment using a public news dataset, trained several GANs architectures to generate synthetic data, and evaluated the performance of several ML models on the augmented dataset. Finally, we modified and extended a data usage approach that was used to evaluate the quality of synthetic data. In this case, we conducted another experiment on a public dataset where we used this framework not only to evaluate the quality of the generated data, but also to understand the relationship between synthetic data quality and data augmentation performance in a classification task. The results indicate that, indeed, the higher the quality of the synthetic data, the better the performance in the underrepresented class.

Resumo

O termo *"fake news"* (expressão em inglês para "notícias falsas") pode ser definido como informação falsa deliberada. Infelizmente, sua proliferação tem vindo a aumentar nos últimos anos. Frequentemente, as *fake news* são utilizadas com o objetivo de obter ganhos políticos ou financeiros, causando problemas sociais severos, tais como a adoção de medicinas alternativas, pontos de vista políticos extremos ou até "esquemas em pirâmide". Para além disso, as redes sociais permitem a sua abrangente e rápida disseminação, tornando fulcral a procura de novos métodos para abordar este problema.

Foram já propostas metodologias de "*Machine Learning (ML)*" (termo em inglês para "aprendizagem computacional") que consistem no treino de modelos com o propósito de identificar *fake news*. Contudo, a maioria dos conjuntos de dados de notícias que estes modelos utilizam para treinar são extremamente desbalanceados, tendo um número substancialmente maior de notícias verdadeiras quando comparado com o número de notícias falsas. Posto isto, os modelos de *ML* acabam por ter dificuldades em generalizar devidamente, impedindo-os de obter resultados satisfatórios.

Uma abordagem promissora consiste em balancear artificialmente o rácio entre notícias falsas e verdadeiras utilizando dados sintéticos. Em situações em que os dados são escassos ou de fraca qualidade, os dados sintéticos podem ser usados para, por exemplo, melhorar a performance de modelos de *ML*. Os dados sintéticos são gerados artificialmente a partir de dados reais e possuem as mesmas propriedades estatísticas que estes. No entanto, ao contrário dos dados reais, que são medidos ou obtidos no mundo real, os dados sintéticos são gerados por algoritmos. Para esse efeito, existem vários métodos, como por exemplo *Random Oversampling (ROS)*, *Synthetic Minority Over-sampling Technique (SMOTE)* ou *Gaussian Mixture Model (GMM)*. Apesar disso, nos últimos anos, um novo modelo generativo tem vindo a ganhar relevância devido aos seus resultados promissores. As *"Generative Adversarial Networks (GANs) "* (termo em inglês para "redes adversárias generativas") são o estado da arte no que toca a modelos generativos profundos, tendo a capacidade de gerar dados sintéticos com a mesma distribuição subjacente que os dados originais. As *GANs* são frequentemente usadas em domínios de relacionados com a imagem, mas ainda não foram suficientemente bem exploradas no domínio dos dados tabulares.

A seguinte dissertação inclui uma revisão da literatura sobre *GANs* (especialmente para dados tabulares), metodologias de geração de dados sintéticos e avaliação de qualidade dos mesmos.

Da pesquisa efetuada, não foram encontradas fontes que combinassem explicitamente estes três tópicos. Após a obtenção destes conhecimentos basilares para este estudo, foi demonstrado o potencial que os dados sintéticos têm para melhorar a performance de modelos de *ML* na deteção de *fake news*. Para além disso, foram analisadas as arquiteturas das *GANs* que se mostraram mais promissoras. Foi realizada uma experiência utilizando um conjunto de dados públicos, e foram geradas amostras sintéticas utilizando *GANs* e avaliando a performance de múltiplos modelos de *ML* no conjunto de dados aumentado. Por último, foi alterado e expandido um método de avaliação de dados sintéticos utilizado para determinar a qualidade das amostras geradas. Para o efeito, foi realizada uma outra experiência, utilizando um conjunto de dados público, onde foi aplicado este método, não apenas para avaliar a qualidade dos dados gerados, mas também para averiguar a relação entre a qualidade dos dados sintéticos e a performance num problema de classificação utilizando o conjunto de dados aumentado. Os resultados indicam que, efetivamente, quanto melhor a qualidade dos dados sintéticos, melhor é a performance na classe sub-representada.

Acknowledgments

As Newton would have put it "If I have seen further it is by standing on the shoulders of Giants". Indeed, I would not have pulled this one off without the help of several people.

I want to start by thanking my beloved parents, Albino Vaz and Natércia Gonçalves, who have supported me my whole life and continue to do so. Their support is, without a doubt, endless.

To my girlfriend, Rita Carneiro, who keeps cheering me up during my (few) existential crises, I express gratitude.

Last but not least, a huge thank you to my supervisor, Professor Álvaro Figueira, who has always encouraged me to go a step further and was continuously present throughout this delightful journey, paddling through uncharted waters and battling sea monsters with me.

What a quest it has been! Thank you!

Contents

List of Tables

List of Figures

Listings

Acronyms

AC-GAN	Auxiliary Classifier Generative Adversarial Network
ADASVM-TW	Adaboost Support Vector Machine Ensemble with time weighting
AE	Autoencoder
AI	Artificial Intelligence
AUROC	Area Under the Receiver Operating Characteristics Curve
BCI	Brain-Computer Interface
BigGAN	Big Generative Adversarial Network
BN	Bayesian Network
CGAN	Conditional Generative Adversarial Network
CoGAN	Coupled Generative Adversarial Networks
CycleGAN	Cycle-Consistent Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
DL	Deep Learning
EEG	Electroencephalography
EQ-GAN	Entangling Quantum GAN
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
InfoGAN	Information Maximizing Generative Adversarial Network
IVE	Immersive Virtual Environments
KL	Kullback–Leibler
MCAE	Multichannel Autoencoder
MetroGAN	Metropolitan GAN
MFC-GAN	Multiple Fake Class GAN
ML	Machine Learning
MuseGAN	Multi-track sequential GAN
PiiGAN	Pluralistic Image Inpainting GAN
ProGAN	Progressive growing of Generative Adversarial Networks
QRAM	Quantum Random Access Memory
ROS	Random Oversampling
RUS	Random Undersampling
SAGAN	Self-Attention Generative Adversarial Networks
SMOTE	Synthetic Minority Over-sampling Technique
SNR	Signal-to-Noise Ratio
StackGAN	Stacked Generative Adversarial Network

StyleGAN	Style-based Generative Adversarial Networks
VAE	Variational Autoencoder
WGAN	Wasserstein Generative Adversarial Networks

Chapter 1

Introduction

The continuous flow of huge amounts of information reaches us constantly through our digital devices. Thus, it becomes more and more difficult to keep up with all this content and to distinguish what is real and what is not.

Fake news are produced for a multitude of purposes, *e.g.*, political or financial gain, and can be roughly defined as intentionally false information [74]. Even though the issue of fake news is not a recent one (the accounts of misinformation, propaganda, and lies can be traced back to as long as $3,000$ years ago [87]), nowadays it is aggravated by online social media platforms, which offer a way to easily and rapidly disseminate fake news. Due to its high reachability and misleading information, spreading large volumes of online fake news can cause severe social problems, such as extreme political standpoints, propagation of pyramid schemes, or the adoption of non-scientific alternative medicines.

A paradigmatic example of the harm caused by fake news is the 2016 US presidential election. According to [6], some commentators believe that former president Donald Trump would not have been elected in the presidential election were it not for the influence of fake news. Indeed, the evidence shows that: (a) 62% of US adults get news on social media; (b) many people who see fake news have reported believing them; (c) the most popular fake news were more widely shared on Facebook than the most popular mainstream news; and (d) Donald Trump tends to be favored over Hillary Clinton in the most discussed fake news [6]. Furthermore, during the presidential race, fake news were employed with the goal of direct political interference, with systematic operations conducted by the Russian government to influence the results of the American election. As shown in [55], Russian efforts to promote fake advertising and videos in favor of Donald Trump, as well as to "provoke and amplify political and social discord in the United States" and the publishment of fake content that had the potential to reach millions of Americans are some examples of how fake news shaped the presidential elections.

Fake news were also present in another important political event, the United Kingdom's referendum decision to leave the European Union, commonly known as "Brexit". In [37], Max Hänska and Stefan Bauchowitz analyzed 7.5 million tweets and found the predominance of

Euroscepticism[1] on social media mirrored its dominance in the press. Moreover, a 2015 Ofcom report found that 43% of people who get news online, receive it through social media and that, the figure rises to 61% among 16- to 24-year-olds (16% of whom rely exclusively on social media for news) [59]. Once again, fake news disseminated through social media platforms played a key role in the final decision.

As shown in the previous paragraphs, the effects of Fake News supported by social media platforms are harmful to society. In fact, an analogy can be made to the disease triangle [72], developed to understand the pathology and epidemiology of plants and their diseases. This model states that three basic elements are required for the manifestation of a disease: the host (in this case, society), the environment (social media), and the infectious agent that carries the virus (Fake News)[2] – see figure 1.1. Overall, Fake News (with the help of social media) causes several societal problems [60].

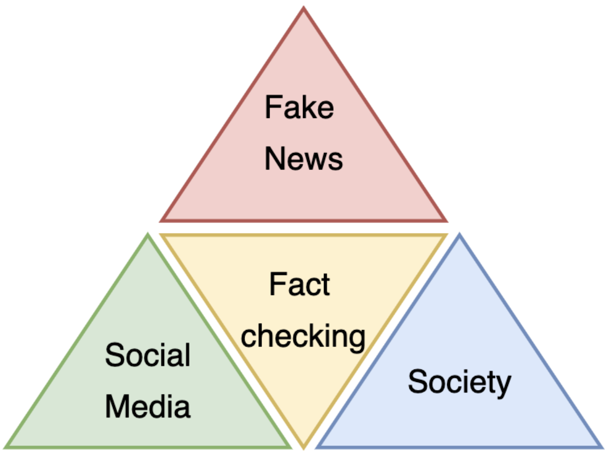


Figure 1.1: Disease triangle. Figure adapted from [60].

The examples presented thus far – 2016 US elections and Brexit – are paradigmatic and effectively capture the harm caused by fake news and, consequently, the urge and great interest to detect when content is unreliable and intended to be deceitful. Given the large amounts of data, it is practically impossible to apply a manual approach, where a human reviewer goes through several pieces of content and classifies them. Hence, Machine Learning (ML) methods, which can automate this process, have been proposed.

Traditional ML algorithms (*e.g.*, Logistic Regression, Naive Bayes, Support Vector Machines (SVMs), or Random Forests) have provided promising results in several studies. Random Forests often figure among the best classifiers, also commonly employed with linguistic-based features, producing good results, such as in [95] or [67]. Other works make use of Deep Learning (DL), since this technique has had good results in natural language processing problems, particularly Recurrent Neural Networks and Convolutional Neural Networks. With the aid of text vector representations and network and semantic features, these networks have shown fine results in [86]

[1]The Euroscepticism is a political position involving criticism of the European Union and European integration.

[2]The analogy between Fake News and a virus is based on the idea that Fake News, like a virus, also requires a host to manifest itself and can only spread through the host itself

and [84].

However, ML models alone are not accurate regarding fake news detection. The problem arises from the fact that news datasets tend to be extremely imbalanced, as the fake news class, typically, has considerably fewer observations than the real news class, posing a problem in fake news detection. Hence, the training of ML models will not be as good as expected since the fake news class is underrepresented. Thus, they will not be able to generalize effectively, leading to a poor classification of unseen observations.

1.1 Motivation

To overcome the lack of fake news labeled samples, most approaches focus on artificially balancing the two classes using oversampling (randomly replicating fake observations), undersampling (removing non-fake examples), or even combining both methods [83]. This methodology is not sufficient to overcome the problem, as randomly adding or removing samples can change the data distribution. A slightly more complex technique for tackling this issue is Synthetic Minority Over-sampling Technique (SMOTE), which creates synthetic samples using nearest neighbors' interpolation. Nonetheless, it is often the case that the result will still have a particular bias due to the lack of training samples, causing this technique not to be capable of significantly improving the ability to generalize.

If the underlying data distribution of fake news can be learned from a small sample, it will be possible to generate new plausible data points which can, in turn, be used by ML models to detect fake news. Hence, a promising solution is to introduce synthetic data. Generative Adversarial Networks (GANs)[3] are a deep generative framework that can comprehend and mimic the data distribution. Hence, they are able to create observations drawn from the learned distribution which are significantly different from the original ones. Moreover, in [83], we have shown some promising initial results. By augmenting our original train set, using a CTGAN (a tabular GAN architecture that will be explored in the following chapters) to generate synthetic data, we were able to increase the performance of some ML models in several metrics.

In [83], we have used an extremely imbalanced fake news dataset in which four different ML models were trained and their performances evaluated with respect to three metrics. Each classifier was trained using the original dataset, but also with the original dataset augmented with synthetic data. The idea was to assess if the models performed better when trained in the augmented datasets, where synthetic observations of the fake news class were added. Indeed, we have seen good and promising results – figure 1.2 shows only the main results of our experiment. The grey lines represent the baseline performances (when the ML models were trained in the original dataset) and the colored lines represent the performance of the models when trained in the augmented datasets (the x-axis shows the number of generated samples). As can be seen, the colored lines surpass the grey ones in most cases, which indicates a performance gain when

[3]GANs will be seen in close detail in the following chapters.

synthetic data is added to the training process[4].

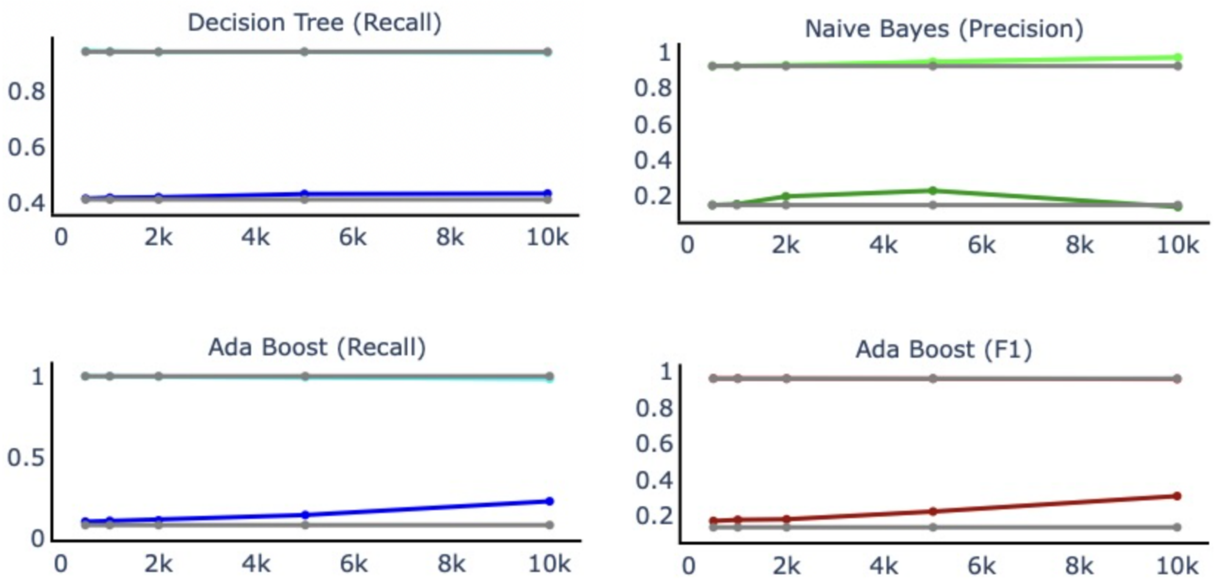


Figure 1.2: Comparison of the performance of the ML models when trained in the original dataset and when trained in the augmented datasets. Image taken from [83].

The experiment described is a first attempt to show that GANs, in this particular case the CTGAN architecture, has the potential to enable MLs models to perform better on classification tasks. Although the results are not the best and can certainly be improved, this experiment has led us to believe that GANs can be quite useful and is worth exploring further.

1.2 Goals

Our main goal is to mitigate the harmful impact of fake news by incorporating synthetic data into the training process of ML models, allowing them to better identify fake news. Consequently, we want to investigate the GAN architectures that offer the best synthetic data for the task at hand. To that end, we propose three different research questions that we attempt to answer in the course of this research:

1. **RQ1: Does the addition of synthetic data in the training process of ML models enables them to better identify fake news?** This is the main research question. Indeed, as the title of this work hints, our main aim is to understand whether synthetic data can help in Fake News identification. As discussed in this chapter, Fake News are compared to an infectious pathogen, and helping to reduce the reach of such pathogen could be an important asset in the fight against Fake News.

[4]For further details of our experiment see [83].

2. **RQ2: Which GAN architectures are more suitable for the generation of tabular data?** This question may help answer the previous one. In fact, by understanding which GAN architectures are more suitable for synthetic data generation, we may transfer that knowledge to the News domain and have more extensive knowledge regarding data augmentation. Answering this question may not only be useful to News domains but to several others where tabular data is involved.

3. **RQ3: How does the intrinsic quality of the synthetic data influence the results of the classification in a data augmented dataset?** This question aims to understand the relationship between the intrinsic quality of synthetic data (i.e., the quality of the data without regard to a specific purpose) and its ability to help in the task of data augmentation. Can generated data that mimic the original very well help in the classification task?

In answering these research questions, this study has a number of related contributions. Firstly, the contents covered in this work offer a strong baseline for any new researcher in the field of synthetic data generation of tabular datasets. Secondly, we use a utility framework to evaluate the quality of synthetic tabular data. Finally, we investigate how the use of synthetic data can aid in the identification of fake news.

1.3 Document Structure

The document is divided into six chapters, including this one, which are organized as follows.

Chapter 2 is dedicated to synthetic data. In section 2.1 an introduction to synthetic data is given, as well as its usefulness. In the next section (section 2.2) we provide a literature review of the main methods used to generate synthetic data, diving them into two main groups — standard and deep learning methods. Whereas in subsection 2.2.1 we showcase the standard methods, in subsection 2.2.2 the deep learning methods are shown. Lastly, in subsection 2.2.3 some thoughts on the algorithms are presented.

Chapter 3 focuses on a particular generative model – GANs. Section 3.1 starts by carefully explaining what is a GAN and how does it generate synthetic data, followed by section 3.2 which contains their main drawbacks. In section 3.3 a literature review of the most prominent GAN architectures is presented by chronological order, so that a clear picture of GAN evolution can be mentally drawn. Finally, section 3.4 is concerned with GANs used for tabular data generation (the main focus of this work) and delves deep into three promising architectures.

Chapter 4 deals with synthetic data quality evaluation. Besides generating synthetic data, it is fundamental to ensure that the data is of proper quality. Section 4.1 shows some important techniques to assess the data quality and, in section 4.2 a critical reflection on the outline methods is provided.

In chapter 5 we present the results from our experiments and how they relate to our research goals. We start by presenting the datasets used (section 5.1) and what GAN architectures were used for the synthetic data generation 5.2. Then we evaluate the quality of the synthetic data (section 5.3) and, finally, we use it to perform data augmentation (section 5.4).

Finally, in Chapter 6 we discuss our main conclusions from the study and answer our research questions. Additionally, we present a general summary of the work conducted (section 6.1), the main contributions (section 6.2), the limitations and challenges faced (section 6.3), and possibilities for extending this work (section 6.4).

Chapter 2

Synthetic Data

This chapter will address synthetic data, what it is and what is used for. Moreover, a literature review of synthetic data generation methods is provided, as well as a critical discussion of the algorithms. The goal is to provide the reader with sufficient context to review the work in the following chapters.

2.1 What is Synthetic Data?

Synthetic data is artificially generated from real data and has the same statistical properties as real data. However, unlike real data, which is measured or collected in the real world, synthetic data is generated by computer algorithms [7, 28].

According to [28], synthetic data can be generated from real data, from existing models, using expert domain knowledge, or from a mixture of these options. Synthetic samples generated from real data are obtained by creating a model that captures the properties (distribution, correlation between variables, etc.) of the real data. Once the model is created, it is used to sample synthetic data.

Synthetic data generated from existing models consist of instances generated from statistical models (mathematical models that have statistical assumptions about how data are generated) or from simulations (*e.g.*, game engines that create images from objects). The use of domain-specific knowledge can also be used to generate synthetic data. For example, knowledge about how the financial market behaves can be used to create an artificial dataset about stock prices. However, this requires extensive knowledge about the domain in question so that the synthetic data behaves similarly to real data.

Many Artificial Intelligence (AI) problems today arise from insufficient, poor quality, or unlabeled data. This is almost ubiquitous, as many fields of study suffer from such difficulties – *e.g.*, physics [3, 75], finance [10], health [24, 46], sports [18], or agriculture [12]. As a result, there is a growing interest in the usefulness of synthetic data and the drawbacks it can overcome.

An example of the usefulness of synthetic data can be found in [82], where a network trained only on synthetic data achieved competitive results when compared to a state-of-the-art network trained on real data. Also, in [56], the author argues that synthetic data are essential for the further development of Deep Learning (DL) and that many more potential use cases remain. He also discusses the three main directions for using synthetic data in Machine Learning (ML): using synthetic data to train ML models and use them to make predictions in real-world data; using synthetic data to augment existing real datasets, typically used to cover underrepresented parts of the data distribution; and solving privacy or legal issues by generating anonymized data. The focus of this work is on the first two directions, to use synthetic data or augmented datasets to enhance the performance of machine learning models, so the generation of anonymized data is not addressed here.

2.2 Synthetic Data Generation Methods

In this section, a review of some of the main methods for generating synthetic data[1] is shown. To better organize them they were divided into standard methods (the most commonly used methods before the success of generative deep learning models) and DL methods, which use DL techniques to generate synthetic data.

2.2.1 Standard Methods

As previously explained, standard methods are the ones that do not use DL techniques to generate synthetic data. As not to randomly present the methods, the section is organized by the level of sophistication of the algorithms. Thus, Random Oversampling (ROS) is shown firstly, followed by Synthetic Minority Over-sampling Technique (SMOTE) and several algorithms that improve the core idea of SMOTE (*e.g.*, by adding safe levels or clustering). Next, cluster-based oversampling is analyzed. Finally, Gaussian Mixture Model (GMM) is reviewed as it provides a different approach to the task of generating synthetic data.

2.2.1.1 ROS

ROS adds additional observations to the dataset by randomly sampling from the minority class(es) with replacement. Probably, the simplest and most straightforward method for expanding a dataset is ROS. Nevertheless, this approach can change the data distribution. Thus, if a classifier is fed with such data, it may learn from an incorrect distribution. Moreover, since ROS duplicates observations, this technique does not create new synthetic samples, but only replicates the existing ones. Therefore, more advanced techniques, such as SMOTE, had to be developed.

[1]Our focus is on tabular data, so we refrain from writing about cropping, zooming, or inverting, which are used in image data.

Examples of ROS can be found, for example, in [13], where the authors compared the use of ROS with Random Undersampling (RUS)[2]. It has been shown that ROS gives better classification results than RUS, since it does not affect the classification of the majority class instances as much as RUS, while it increases the classification of the minority class instances. Another example is shown in [27] where the authors also compare ROS and RUS. In that study, however, they concluded that ROS was surprisingly ineffective, producing little or no change in classification performance in most cases.

2.2.1.2 SMOTE

SMOTE [17] is an oversampling approach in which synthetic observations are generated (and not duplicated, as in ROS) from the minority class(es). SMOTE was inspired by a perturbation method used to recognize handwritten digits. This was a very domain-specific problem and so were the techniques used (*e.g.*, rotation or skew), but the authors of SMOTE generalized them to generate synthetic samples in a less application-specific way.

The algorithm works as follows. Given a data point from a minority class and its nearest neighbor from the same class, the distance between them is determined (the distance is computed as the difference between both feature vectors, the data points). This distance is multiplied by a random number between 0 and 1 and added to the selected data point. This causes the new sample to fall in the line segment between the original sample and its neighbor. The same process is repeated until the desired number of samples is reached. Figure 2.1 shows a toy example of an iteration of the SMOTE algorithm.

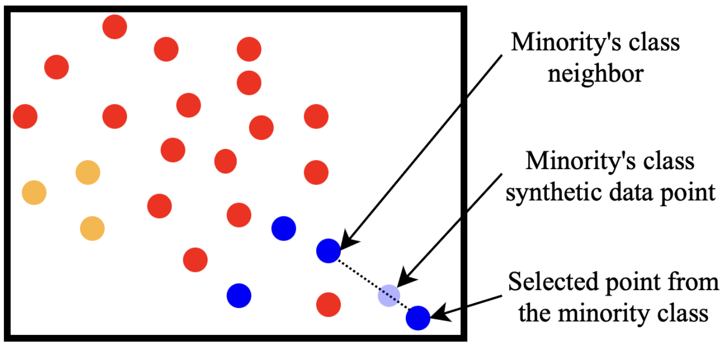

Minority's class neighbor
Minority's class synthetic data point
Selected point from the minority class

Figure 2.1: Toy example of the SMOTE algorithm for one iteration. The dataset has 2 minority classes (blue and orange), and one majority class (red). After selecting a minority class instance and its nearest neighbor, a synthetic data point is added somewhere in the line segment between them.

The SMOTE algorithm is quite popular in the literature. In [51], for example, the authors evaluate the use of SMOTE for high-dimensional data. It was shown that SMOTE does not attenuate the bias toward the majority class for most classifiers. However, for k-nearest neighbors,

[2]RUS is a technique that consists of randomly removing instances of the majority class so that minority classes are not underrepresented.

classifiers based on Euclidean distance, SMOTE may be beneficial if the number of variables is reduced by variable selection. In [79], SMOTE is combined with decision trees and bagging to address a problem of imbalanced credit evaluation of companies. The proposed framework shows good results, outperforming other five different approaches, and overcoming the class imbalance problem. Another example of using SMOTE can be seen in [80], where SMOTE is combined with Adaboost Support Vector Machine Ensemble with time weighting (ADASVM-TW) in two different ways and evaluated in a financial dataset. The first method uses SMOTE followed by ADASVM-TW, while the second method embeds SMOTE into the iteration of ADASVM-TW. Both approaches greatly improved the recognition performance of the minority class.

Although a more advanced technique than ROS, SMOTE still suffers from some problems – *e.g.*, focusing on minority class instances (thus ignoring those of the majority class), or altering the true data distribution. That being said, some informed improvements can be applied. Therefore, Borderline-SMOTE, Safe-Level-SMOTE, and ADASYN have been introduced.

2.2.1.3 Borderline-SMOTE

Han *et al.* [36] have proposed two algorithms that are a variation of SMOTE: Borderline-SMOTE1, which only oversamples the minority class(es) examples near the borderlines, and Borderline-SMOTE2, which takes also into account the majority class observations.

The Borderline-SMOTE1 considers only the minority class data points that have a number of minority class neighbors in the range $[m/2, m[$, where m is defined by the user. These are the points that can be easily misclassified (the borderline data points of the minority class). After detecting such observations, SMOTE is applied to create new synthetic samples. Borderline-SMOTE2 is quite similar, with the difference that it also considers the neighbors of the majority class. According to [36], Borderline-SMOTE offers improvements over ROS and SMOTE in terms of TP-rate and F-value.

Examples of Borderline-SMOTE can be found in [48], where the authors use this method for data augmentation and evaluate its impact on an Electroencephalography (EEG) classification dataset obtained with a Brain-Computer Interface (BCI). Borderline-SMOTE did not improve overall classification performance, but significantly increased the performance of the classifiers that produced the worst results. Another example can be found in [68], where Borderline-SMOTE was improved by using Gabriel graphs. The authors addressed the main problems of Borderline-SMOTE and were able to improve its performance on neural networks.

2.2.1.4 Safe-Level-SMOTE

SMOTE synthesizes minority class samples along a line connecting a minority class instance to its nearest neighbors, ignoring nearby majority class instances. Safe-Level-SMOTE [15], on the other hand, defines safe regions to prevent oversampling in overlapping or noisy regions, providing better accuracy performance than SMOTE and Borderline-SMOTE.

Each minority class example is assigned a safety level defined as the number of instances of the minority class in the k nearest neighbors, where k is specified by the user. Each synthetic instance is positioned closer to the largest safe level so that all synthetic instances are created only in safe regions. Intuitively, when given a data point, p, from the minority class and its nearest neighbor, n, (from that same class), the Safe-Level-SMOTE will generate a synthetic sample closer to p if its safe level is higher than the one of n, or closer to n otherwise. That is, the synthetic sample will be closer to the data point that has more nearest neighbors from the minority class. Hence, the Safe-Level-SMOTE offers a wittier solution than the one of SMOTE, in the sense that it does not simply generate a random instance in the line segment between two minority class data points, but takes into account their neighborhoods.

An example of using Safe-Level-SMOTE is shown in [76], where the authors overcome some of the difficulties of Safe-Level-SMOTE (some synthetic data points may be placed too close to nearby majority instances, which can confuse some classifiers and also the fact that it avoids using minority outcast samples for generating synthetic instances) by using two processes – moving the synthetic instances of the minority class away from the surrounding examples of the majority class and treating the outcasts of the minority class with a 1-nearest-neighbor model. Several ML models were evaluated with 9 UCI and 5 PROMISE datasets after using the above approach, and improvements in F-measure were obtained.

2.2.1.5 ADASYN

ADASYN [38] is an oversampling algorithm that improves the learning performance of the classifiers. It uses a weighted distribution for different minority class instances that takes into account their level of difficulty for a classifier to learn – the minority class samples that have fewer minority class neighbors are harder to learn than those which have more neighbors of the same class. Thus, more synthetic samples are generated for the minority class examples that are harder to learn and less for the minority class examples that are easier to learn.

ADASYN is similar to SMOTE in the sense that it generates synthetic samples in the line segments between two minority class data points. The difference is that ADASYN uses a density distribution as a criterion to automatically determine the number of synthetic samples to generate for each instance of the minority class. Hence, the extended dataset provides a balanced representation of the data distribution and forces the classifier to pay more attention to the more difficult-to-learn examples.

The ADASYN approach is used in [50], to process an imbalanced telecommunications fraud dataset. The authors concluded that ADASYN is more beneficial than SMOTE and that accuracy, recall, and F1-measure were improved when ADASYN was used. Another example can be found in [1], where ADASYN is used this time for data augmentation in an imbalanced churn dataset. A final example is retrieved from [19], where ADASYN is used in a financial dataset. The authors note that ADASYN overcame the problem of overfitting caused by SMOTE and improved the prediction of extreme financial risk.

While ADASYN, Safe-Level- and Borderline-SMOTE are variants of SMOTE, it is also possible not to modify the SMOTE algorithm, but instead, use an unsupervised algorithm before performing SMOTE (or ROS). Clustering algorithms are a type of unsupervised algorithm that can be very useful in detecting structure in the data (*e.g.*, divide the data into classes). When applied well, clustering algorithms can reveal hidden patterns in the dataset that were previously undetectable.

2.2.1.6 K-Means SMOTE

K-Means SMOTE was proposed by Last, Douzas, and Bacao in [26], and combines K-means [52], a popular clustering algorithm, with SMOTE, thereby avoiding the generation of noise and effectively overcoming the imbalances between and within classes.

K-Means SMOTE consists of three steps. First, observations are clustered using the K-means algorithm. This is followed by a filtering step in which the clusters with a small proportion of minority class instances are discarded. The number of synthetic samples to be created also depends on the cluster. That is, clusters with a lower proportion of minority class samples will have more synthesized instances. Finally, the SMOTE algorithm is applied to each of the clusters. Figure 2.2 shows the use of K-Means SMOTE in a toy dataset.

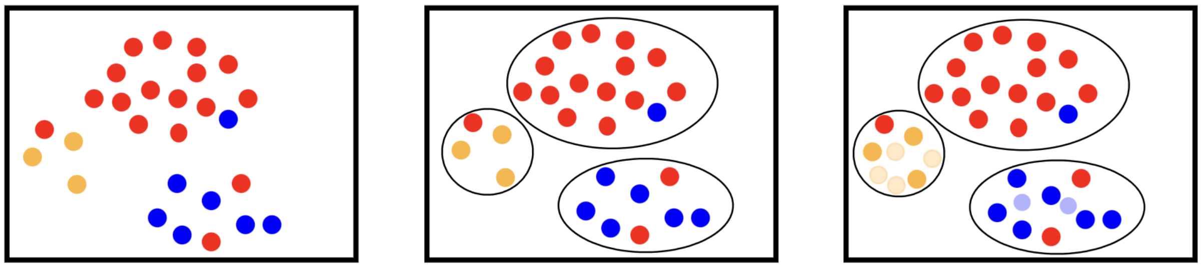


Figure 2.2: Toy example of the K-Means SMOTE algorithm. The left image shows the toy dataset, which consists of 3 classes: the blue and the orange are minority classes and the red one is a majority class. At the center took place the creation of clusters. In the right picture, the clusters with a high proportion of samples from the minority class were populated with synthetic instances.

An example of the use of K-Means SMOTE can be found in [71], where the authors compared it with other methods of generating synthetic data, such as SMOTE or Borderline-SMOTE. It was shown that K-Means SMOTE is better at balancing datasets allowing ML models to perform better in terms of average recall, F1-score, and geometric mean.

2.2.1.7 Cluster-Based Oversampling

Jo and Japkowicz, in [42], address the presence of small disjuncts in the training data. Their work has shown that the loss of performance in standard classifiers is not caused by class imbalance, but that class imbalance can lead to small disjuncts, which in turn cause the loss of performance.

The Cluster-Based Oversampling algorithm consists of clustering the data for each class, *i.e.*, each class is clustered separately (in [42] the authors used K-Means clustering, but theoretically any clustering algorithm can be used), and then applying ROS to each cluster. For the majority class clusters, all clusters except the largest are randomly oversampled until they have the same number of observations as the majority class cluster with the most data points. The minority class clusters are randomly oversampled until each cluster has m/N samples, where m is the number of instances of the majority class (after ROS) and N is the number of clusters of the minority class.

Cluster-Based Oversampling is similar to K-Means SMOTE, in that both use clustering followed by oversampling, but they differ in some aspects. For instance, K-Means SMOTE uses a specific clustering algorithm, K-Means, and the classes are not clustered separately, while Cluster-Based Oversampling allows the user to freely choose the clustering algorithm and the classes are clustered separately. Also, K-Means Clustering uses SMOTE, while Cluster-Based Oversampling uses ROS.

The methods studied so far, with the exception of ADASYN, tend to neglect the distribution of the original data. Thus, a logical but different approach would be to model the underlying distribution of the data and draw a sample from it. However, estimating such a distribution is an extremely difficult problem, especially as the number of features in the data increase and simplifications need to be made.

2.2.1.8 GMM

A GMM is a probabilistic model that assumes that the data can be modeled by a weighted sum of a finite number of Gaussian distributions [47]. So, the resulting model is given by

$$p(x) = \pi_1 p_1(x) + \pi_2 p_2(x) + ... + \pi_n p_n(x)$$

In the univariate case, $p_i(x)$ is the probability density function of a univariate normal distribution with mean μ_i and standard deviation σ_i, π_i is the weight assigned to each $p_i(x)$, and n is the number of components. The number of components, n is set by the user, and the parameters $\mu_1, \sigma_1, \mu_2, \sigma_2, ..., \mu_n, \sigma_n$ and $\pi_1, \pi_2, ..., \pi_{n-1}$[3] are estimated, typically by an expectation-maximization algorithm – an iterative and well-founded statistical algorithm that calculates the

[3]The sum of all π_i equals 1, so if $n-1$ weights are estimated, the last one is equal to 1 minus their sum. That is, $\pi_j = \sum_{i \neq j}^{n} \pi_i$

probability that each point is generated by each component, and then changes the parameters to maximize the likelihood of the data. For the multivariate case, $p_i(x)$ is replaced by a multivariate normal distribution, $N_k(\mu_i, \Sigma_i)$, where k is the dimension of the multivariate normal distribution, μ_i is now a vector of means, and Σ_i is the covariance matrix. Having determined the model, synthetic data are generated by drawing random samples from it.

An example of using GMM can be found in [22]. The authors address the problem of lack of data in Immersive Virtual Environments (IVE) by using a GMM. The results have shown that the GMM is a good option to overcome the problem of small sample size in IVE experiments.

2.2.2 Deep Learning methods

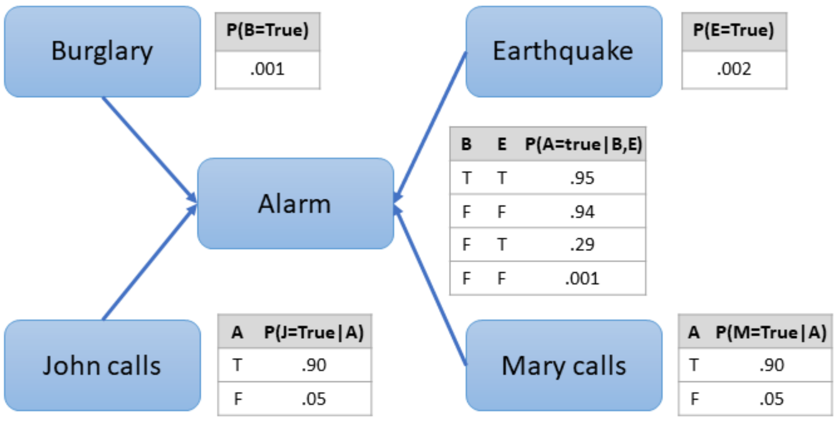
DL methods are so named because they use DL techniques to create new instances. Unlike standard methods, DL models are more difficult to understand because they are more complex and usually cannot be interpreted. In this section, we review the three main classes of deep generative models: Bayesian Networks (BNs)[4], Autoencoders (AEs), and Generative Adversarial Networks (GANs). There are innumerable variations of these algorithms and a whole range of domain-specific architectures. It would, therefore, not be possible to list everything in the literature, so instead a comprehensive overview is presented.

2.2.2.1 Bayesian Networks

A BN (also known as a belief network in the 1980s and 1990s) is a type of probabilistic graphical model that uses Bayesian inference for probability computations over a directed acyclic graph [77]. It is used to represent dependence between variables so that any full joint probability distribution can be represented, and in many cases very succinctly [69]. In a BN, each node corresponds to a random variable (which may be discrete or continuous) and contains probability information that quantifies the effect of the parents (the nodes pointing to it) on the node. If there is a link from node x_i to node x_j, then x_i has a direct impact on x_j. Moreover, if there is a path from node x_i to node x_j (with at least one node in between), then x_i also has an influence on x_j (though not a direct influence).

As an example, suppose a certain person has an alarm system installed at home. The alarm is very good at detecting burglaries, but it also triggers for minor earthquakes. The person has asked his neighbors, John and Mary, to call him if the alarm goes off. On the one hand, however, John is more careful than Mary, so he almost always calls when he hears the alarm, but sometimes mistakes it for the phone ringing. On the other hand, Mary likes to listen to loud music, so she does not hear the alarm as often as John does. This is a simple toy example that can be modeled by a BN (see figure 2.3).

[4]Even though BNs may not be considered DL, they are easily generalized to Bayesian Neural Networks, which are DL structures [11], so we have included them.



P(B=True)
.001

P(E=True)
.002

B	E	P(A=true|B,E)
T	T	.95
F	F	.94
F	T	.29
F	F	.001

A	P(J=True|A)
T	.90
F	.05

A	P(M=True|A)
T	.90
F	.05

Figure 2.3: Toy example of a BN. A certain individual has an alarm installed at home, which fires in case of minor earthquakes or burglaries (with a certain probability). The given individual also has two neighbors, Mary and John, who will call him in case they hear the alarm. Image adapted from [69].

The previous example is quite simple, but these structures can have many more layers, representing dependencies between multiple variables. As the number of layers increases, BNs become deep BNs. Although they played an important role in the history of DL[5], they are rarely used nowadays [34].

An example of the use of BNs can be seen in [92], where the authors address the problem of sharing private data. A BN adds noise to the data to estimate an approximate distribution of the original data. The BN has been evaluated experimentally and found to significantly outperform existing solutions in terms of accuracy.

2.2.2.2 Autoencoders

An AE is a special type of feedforward neural network that consists of two parts: an encoder network that learns to compress high-dimensional data into a low-dimensional, latent spacial representation (the code), and a decoder network that decompresses the compressed representation into the original domain [32]. Figure 2.4 shows a diagram of an AE.

Formally, the encoder can be viewed as a function, $c = E(x)$, that produces a low-dimensional representation of the data, and the decoder as a function, $r = D(c)$, that produces a reconstruction of the code. The goal is not for the AE to learn how to set $D(E(x)) = x$ for each input example x, but rather to learn how to copy the original data only approximately, and only inputs that resemble the original data. By constraining it and forcing it to learn which aspects of the data should be prioritized, AEs can learn useful properties about the data[6] [35].

[5] BNs were one of the first non-convolutional models to allow training of deep architectures successfully.

[6] AEs have been on the DL landscape for decades, and have typically been used for feature learning and dimensionality reduction.

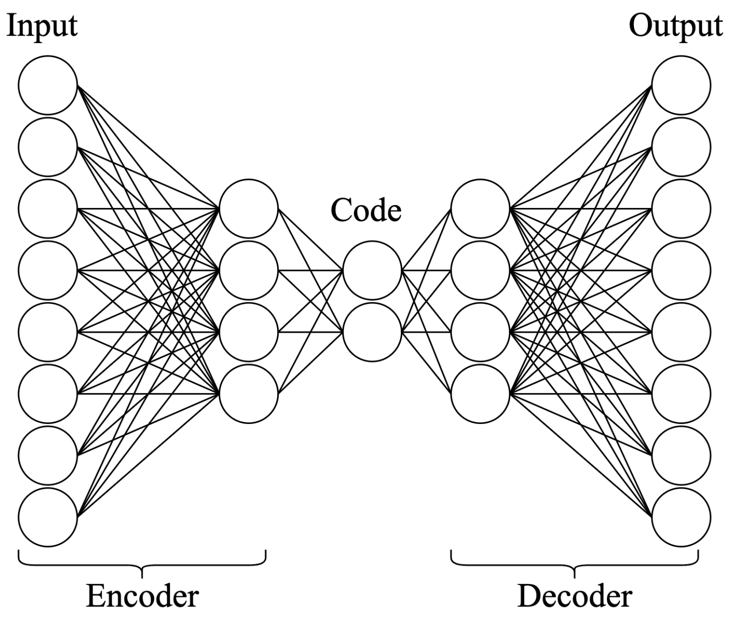


Figure 2.4: A diagram of an AE. On the left is depicted an example structure of an encoder. The input layer has more units than the middle layer, where the input was compressed into a lower-dimensional representation (the code). On the right, the decoder decompresses the code back to the original domain.

In terms of generating synthetic samples, AEs have some issues. First, the learned distribution of points in the latent space is undefined, *i.e.*, when a point is sampled from the latent space and decoded to generate a new example, there is no guarantee that it is a plausible sample. Second, there is a lack of diversity in the generated samples. Finally, points belonging to the same class may have large gaps in the latent space, which can lead to poorly generated instances when samples are drawn from their neighborhood. To overcome these problems, Variational Autoencoders (VAEs) can be used.

VAEs were first proposed by Diederik P Kingma and Max Welling, in [45], and are a natural extension of AEs aimed at solving the aforementioned problems. VAEs improve on vanilla AEs by making a few changes to the encoder and loss function:

Encoder. The encoder of a VAE maps each point in the original data to a multivariate normal distribution in the latent space, represented by the mean and variance vectors. VAEs assume that there is no correlation between any two dimensions in the latent space, so the covariance matrix does not need to be calculated because it is diagonal. This small change ensures that a point, a, sampled from the neighborhood of another point, b, in the latent space, is similar to b. Thus, a point in the latent space that is completely new to the decoder will most likely still yield a correct sample.

Loss function. The VAE loss function adds the Kullback–Leibler (KL) divergence to the AE reconstruction function (typically, the binary cross entropy or the root mean squared error). Formally, the KL divergence in this particular case can be written as follows.

$$KL\Big(N(\mu,\sigma) \parallel N(0,1)\Big) = \frac{1}{2}\sum_{i=1}^{k}(1 + log(\sigma_i^2) - \mu_i^2 - \sigma_i^2)$$

where k is the number of dimensions in the latent space. So, the loss function becomes

$$L(x,\hat{x}) = RL(x,\hat{x}) + \frac{1}{2}\sum_{i=1}^{k}(1 + log(\sigma_i^2) - \mu_i^2 - \sigma_i^2)$$

where RL is the reconstruction loss, x denotes the input data, and $\hat{x}$ is the predicted output. This loss function provides a well-defined distribution (the standard normal distribution) that can be used to sample points in the latent space – sampling from this distribution most likely guarantees that the sampled points are in the region from which the decoder is to decompress. Also, the gaps between points in the latent space will be smaller.

Therefore, changing the encoder mapping and adding the KL divergence to the loss function leads to a better framework for generating synthetic samples – the VAE.

The use of AEs to generate synthetic data is widespread in the literature. For example, in [94], the authors used a Multichannel Autoencoder (MCAE) to assist classifiers in the learning process. They concluded that the use of a MCAE provided better feature representation. In addition, the experimental results validated their methodology for generating synthetic data. In [85], a VAE was used to address a problem of imbalanced image learning. It was shown that the VAE can generate novel samples and that it produces better results compared to other methods in several distinct datasets with different evaluation metrics. A final example of the use of AEs can be seen in [41], where a VAE was used to generate accident data, which was then used for data augmentation. The VAE was compared to SMOTE and ADASYN. This showed its superiority as it provided a better learning process for the classifiers, and thus provided better classification metrics.

2.2.2.3 Generative Adversarial Networks

GANs are a type of deep generative structure consisting of two networks: the generator, G, and the discriminator, D. The details of how they operate will be shown in fine detail in Chapter 3, and for now, we will focus on the practical applications of such structures. Due to the usefulness of GANs in generating synthetic samples, they are widely used. Hence, it would be tedious to list them all. Therefore, only some interesting results will be shown.

In [30], the authors used a GAN to generate artificial EEG datasets. The results presented were quite good: indeed, GANs (in this case, with convolutional layers) were able to generate brain signals similar to the real ones (obtained by EEG in multiple subjects).

Patel *et al.* used a Conditional Generative Adversarial Network (CGAN) for data augmentation in a signal modulation dataset used for automatic modulation classification [63]. These data were then used to improve the accuracy of a CNN classifier used as a benchmark. It was concluded that CGAN-enriched data could greatly benefit CNN-based training – it has faster convergence and lower training loss. Moreover, the more data generated by the CGAN, the better the F1-score of the CNN classifier is (the authors used 1000, 2000, 3000, 4000, and 5000 synthesized samples). Figure 2.5 shows the F1-score for the original and the extended dataset at different Signal-to-Noise Ratios (SNRs). Clearly, the F1-score increases at each SNR level as more synthetic samples are added to the original dataset.

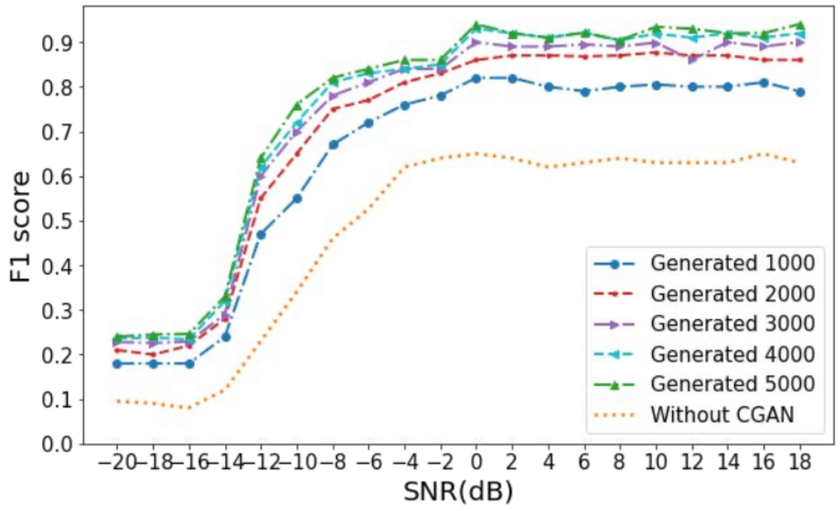


Figure 2.5: F1-score on the original data and on the augmented datasets (1000, 2000, 3000, 4000, and 5000 synthetic samples were added to the original data) at different SNR levels. The plot shows that, as the number of generated samples increases, the better the F1-score at each SNR level. Image taken from [63].

Another example of the use of GANs is the Multiple Fake Class GAN (MFC-GAN) [4]. The MFC-GAN was used to handle datasets with multiple imbalanced classes by augmenting the original data with artificial samples. Four public datasets were used, MNIST, E-MNIST, CIFAR-10, and SVHN, and MFC-GAN was compared with FSC-GAN [5], Auxiliary Classifier Generative Adversarial Network (AC-GAN) [58], and SMOTE [17], both in terms of the quality of the generated samples and in a classification task (a baseline CNN classifier was used). It was found that MFC-GAN provided better quality generated samples and that the training time was significantly reduced compared to FSC-GAN (MNIST dataset). The results also showed that MFC-GAN performed better than SMOTE and AC-GAN on all SVHN and CIFAR-10 minority classes and in 7 of 10 E-MNIST and MNIST minority classes.

In [81], Sushko *et al.* proposed the One-Shot GAN, which given just one image (or video) as input, can generate images (or videos) that are significantly different from the original one. This type of GAN has improved the quality and variety of images (and videos) over previous works when only one image (or video) is available. When only small amounts of data are available, the One-Shot GAN mitigates the memorization problem (reproducing the original image) and is able to generate images that are structurally different from the original. This is extremely useful for data augmentation tasks in domains where data is very scarce and collecting it may be challenging.

A quantum GAN – Entangling Quantum GAN (EQ-GAN) – was proposed in [57]. By leveraging quantum circuits' entangling[7] power, it overcomes some limitations of previously proposed quantum GANs (non-convergence due to mode collapse and a non-unique Nash equilibrium). Moreover, the authors have shown that the EQ-GAN can generate an approximate Quantum Random Access Memory (QRAM), which is required by most ML algorithms. They have further demonstrated an application of such a QRAM, improving the performance of a quantum neural network in a classification task.

Finally, to conclude this subsection, we show one last example. In [93], the authors have proposed the Metropolitan GAN (MetroGAN), which is used for urban morphology simulations. Recent studies have shown that GANs have the potential to simulate urban morphology, despite being a challenging task. Nevertheless, the existing GAN models are limited by the instability in model training and the sparsity of urban data, compromising their application. However, when compared to other state-of-the-art urban simulation methods – XGBoost, U-NET, and CityGAN – the MetroGAN outperforms them all in the three levels used to evaluate the results: pixel level, multi-scale spatial level, and perceptual level.

2.2.3 Thoughts on the Algorithms

In this chapter, eight techniques for data augmentation were reviewed. ROS is the most simple of them all and, therefore, is easier to implement than any of the others. However, it is a very naive approach that does not take into account the distribution of the data. Furthermore, it disregards the majority class instances, as well as the difficulty of the classifiers in learning the decision boundaries. A simple yet more intelligent way to improve ROS is SMOTE. This technique does not replicate observations as ROS does but adds new synthetic data points to the dataset. This can make it easier for a classifier to learn from the data. Nonetheless, SMOTE does not care about changing the distribution of the data and does not consider majority class observations.

SMOTE brought a highly successful synthetic data generation method but also a lot of room for improvement. Therefore, new algorithms were created by borrowing the core idea of SMOTE, which is to add noise to the instances. Borderline-SMOTE oversamples near the borderlines to make the learning task easier for classifiers while also taking into account the majority class observations. Safe-Level-SMOTE has defined safe regions to generate better quality instances, which is an improvement over SMOTE and Borderline-SMOTE.

K-Means SMOTE first clusters the data using the K-Means algorithm and then oversamples the clusters using SMOTE, effectively overcoming the imbalances between and within classes. ADASYN is another variant of SMOTE. This method takes into account the learning difficulties

[7]Quantum entanglement is a physical phenomenon that happens when, in a set of particles, an individual particle's quantum state cannot be described independently of the state of the others, no matter how far apart they are.

of the classifiers and aims not to change the data distribution (one of the drawbacks of SMOTE). Cluster-Based Oversampling takes into account the presence of small disjuncts in the data. This algorithm is not a variant of SMOTE but a variant of ROS. Both the minority and majority classes are oversampled so that each class has the same number of instances.

GMMs use a different approach to address the synthetic data generation task – modeling the data with a weighted sum of normal distributions. While this is usually an improvement over previous algorithms, it has two major drawbacks. First, not all datasets can be modeled with a weighted sum of the Gaussian distribution. Therefore, the use of GMM may not be the most appropriate method for generating plausible samples. On the other hand, some types of data may have categorical features. In these cases, GMM cannot be applied because the normal distribution is continuous, and it cannot model discrete variables.

BNs, AEs, and GANs are more complex techniques compared to the others. Unlike the previous methods, they use a DL approach that allows them to better learn the underlying patterns in the data and, therefore, offer higher quality synthetic patterns in most cases. BNs were widely used in the past but have fallen out of favor and are rarely used today. AEs, especially VAEs, are powerful generative models that have evolved and are proving useful in data generation tasks.

Nevertheless, AEs are not as popular and usually not as powerful as GANs. Yann LeCun has even described them as "the most interesting idea in the last 10 years in machine learning" [53]. GANs have countless different architectures, and many are yet to be created. Only a few applications of GANs for the generation of samples were shown, as it would be grueling (and probably impossible) to find and summarize all the literature on GANs and data generation. They can be quite problem-specific, so a few have been selected to show their capabilities and broad application to real-world data.

In the following chapter (Chapter 3) we will dive into the details of how GANs work, their main drawbacks, and several GAN architectures that have been developed in the last few years. As seen in the present chapter, they are a powerful tool to generate synthetic samples, with high applicability and, thus, deserve to be further explored.

Chapter 3

Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a framework that uses an adversarial process to estimate generative deep learning models, proposed by Ian J. Goodfellow *et al.* [33] in 2014. These structures have been adapted and improved over the last years and are now very powerful. GANs are currently capable of painting, writing, composing, and playing, as we will see in this chapter.

3.1 GANs Under the Hood

A GAN is constituted by two models[1]: a generator model, G, that tries to generate samples that follow the underlying distribution of the data. Nonetheless, these observations are suitably different from the ones in the dataset (*i.e.*, they should not simply reproduce observations that already occur in the dataset). There is also a discriminator model, D, that given an observation (from the original dataset or synthesized by the generator), classifies it as fake (produced by the generator) or real. An important thing to consider is that G and D compete against each other. While G generates similar data points to those in the original data, with the aim of deceiving the discriminator, D attempts to distinguish the generated from the real observations.

To describe in closer detail how the networks are trained, the training was split into the training of the discriminator and of the generator separately. Training the discriminator consists in creating a dataset with instances generated by G and data points from the original dataset. The discriminator outputs a probability (continuous value between 0 and 1) that indicates whether a given observation came from the original data (0 means that the discriminator is 100% certain that the given example was synthesized, while 1 means the exact opposite).

[1]Typically, the models used for the generator and discriminator are neural networks. As such, we normally refer to G and D as networks. However, in theory, the models need not to be a neural network. Indeed, in [33], the term "model" is used. Nonetheless, they note that the "adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons".

The training of the generator is more complicated. G is given as input random noise[2], commonly from a multivariate normal distribution, and the output is a data point with the same features of the original dataset. However, there is no dataset to inform whether a particular point in the latent space is mapped by G into a reasonable or useful example. Therefore, the generator is only provided with a value from a loss function. This is usually the binary cross-entropy[3] between D's output and a response vector of 1's (the instances synthesized by G are all marked as coming from the original data).

Given the discriminator's feedback, *i.e.*, the value of the loss function, the generator attempts to improve to better fool the discriminator. As training progresses, G uses D's output to generate better examples, *i.e.*, examples that better resemble the real data distribution. As the data produced by G becomes more realistic, D also improves, so that it can better determine whether a sample is real or synthetic. As such, both networks improve each other and, ideally, G will be able to mimic the data distribution and D will be $\frac{1}{2}$ everywhere, *i.e.*, the probability that D distinguishes between a real observation and a generated one is as good as a random guess. In this ideal scenario, G has succeeded in recovering the distribution of the original data, completely fooling D. A GAN diagram is shown in Figure 3.1.

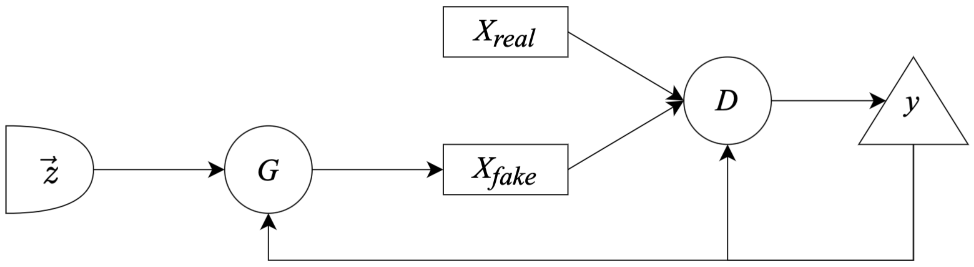


Figure 3.1: A GAN diagram. G is given random noise $\vec{z}$, usually from a multivariate normal distribution, to generate a set of data points, X_{fake}. D is provided with both the original data, X_{real}, and the generated data. D outputs a label y, denoting if a given observation is fake (was produced by G) or real (came from the original data).

3.2 Main Drawbacks

Although plain vanilla GANs – that is, the GANs in their simplest form, as we have been explaining – are quite strong ideas, they also have disadvantages. Namely, GANs are extremely

[2]The term *latent space* is typically used to designate G's input space.

[3]The binary cross-entropy is mathematically defined as follows

$$-\frac{1}{n}\sum_{i=1}^{n} y_i log(p_i) + (1 - y_i) log(1 - p_i)$$

where y_i represents the label of an input sample, p_i is the probability of y_i coming from the original data, and n is the number of examples. It is a measure of the difference between the ground truth and the computed probabilities and it is used in the case where there are only two possible outcomes – the observation came from the original data or it was synthesized by the generator.

difficult to train due to a number of factors that include the loss function, hyperparameters, or a generator that can easily fool the discriminator.

Oscillatory loss (instability) is a common problem that occurs during the training process. It is characterized by wild oscillations of the discriminator's and generator's loss, which should be stable over the long term. For the training process to be effective, the loss should stabilize or gradually increase/decrease over the long term. Unfortunately, in many cases, this is not what happens. Another problem with the loss function is the lack of information it usually provides (uninformative loss). For example, a commonly used generator's loss function is the binary cross entropy. This is a disadvantage because there is no correlation between the generator's loss and the quality of the output (not only in the specific case of the binary cross entropy). Hence, the training is sometimes difficult to monitor.

Another fairly common phenomenon is that the generator finds a small number of samples that fool the discriminator – this is called mode collapse. Having found such samples, the generator will focus only on them to minimize its loss function, while the discriminator remains confused during training because it cannot distinguish whether the instances are real or synthetic. Therefore, the generator is not able to produce other examples than this limited set. Figure 3.2 shows an example of mode collapse in a toy dataset.

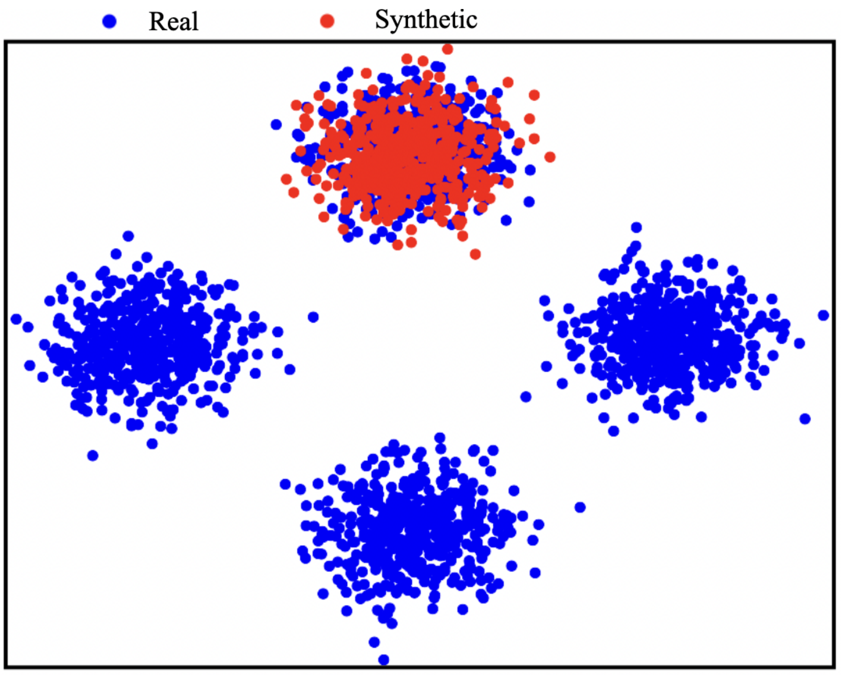


Figure 3.2: Graphical representation of mode collapse in a toy dataset consisting of random samples drawn from four Gaussian distributions with the same covariance matrix but different means (visible by the four separate clusters). The blue points correspond to real data points, whereas the red ones are synthesized by the generator. The generator has found a small number of samples (the ones in the upper cluster), so it does not learn beyond that. It will continue to produce samples in that range without seeing the overall distribution of the data, as it is enough to fool the discriminator.

Moreover, GANs have a significant number of hyperparameters. Thus, to create a well-performing GAN, a large number of hyperparameters must be tuned. It is possible to use grid search, but only for a limited subset of hyperparameters. Otherwise, the training time will be considerably long and the resource consumption extremely high.

Finally, there is the vanishing gradient problem, which may completely stop the GAN from further training, given that the gradients can be extremely small and not allow the weights to be updated further. This can occur if the discriminator is close to optimal, which allows it to accurately discern generated samples from real ones and causes the generator's train to fail.

These are the five most common problems encountered in GAN training – oscillating loss, mode collapse, uninformative loss, vanishing gradients, and hyperparameter-tuning. The above problems are broad and independent of domain and architecture. That is, they attempt to cover the range of possible GAN training drawbacks without being too specific about the loss function or hyperparameters used (broad); they do not depend on the particular domain, whether it is live cell images or a tabular dataset of bank fraud (domain-agnostic); and finally, they do not depend on a particular GAN architecture (architecture-agnostic).

3.3 GANs Come in a Lot of Flavours

Since GANs were proposed, many researchers have considered them a powerful tool. As a result, they have been systematically modified and improved. The architecture of a GAN can be very problem-specific, and they are often modified or fine-tuned to serve a particular purpose. Hence, the literature on them is quite extensive, and thus, only the main highlights are shown. In the following paragraphs, the GAN architectures are arranged chronologically by year (in ascending order, *i.e.*, earlier years are shown first), so two architectures created in the same year may not be arranged by month. Nonetheless, this can show the evolution of GANs up to the time of writing (September 2022).

Conditional Generative Adversarial Network (CGAN) is a GAN variant proposed by Mehdi Mirza and Simon Osindero in [54]. Suppose one is using a vanilla GAN on an image dataset with multiple class labels, (*e.g.*, the ImageNet dataset). The GAN has been properly trained and is ready to generate synthetic samples. However, it cannot sample an image of the desired class. For example, if one wants synthetic images of cars (assuming that images of cars were used in the training data), one cannot force a vanilla GAN to do so. This happens because there is no control over the latent space representation. That is, the GAN maps points from latent space to the original domain, but the features in the latent space are not interpretable by the user. As such, one does not know from which range of points to sample in order to produce examples of a certain class. This is an obvious disadvantage of using GANs in labeled datasets. An interesting idea is to make the GAN dependent on a class label, which allows generated data to be conditioned on class labels. That is, given a labeled dataset, the CGAN is trained using the data instances and their respective labels. Once trained, the model can generate examples that depend on a class label selected by the user. For example, if a hypothetical dataset has three classes – “low”, “medium”, “high” – the CGAN is trained with both the instances and their associated labels. After the learning process is complete, the user can choose to generate samples of only “low” and “high” classes by specifying the desired label. A diagram representing a CGAN is shown in Figure 3.3.

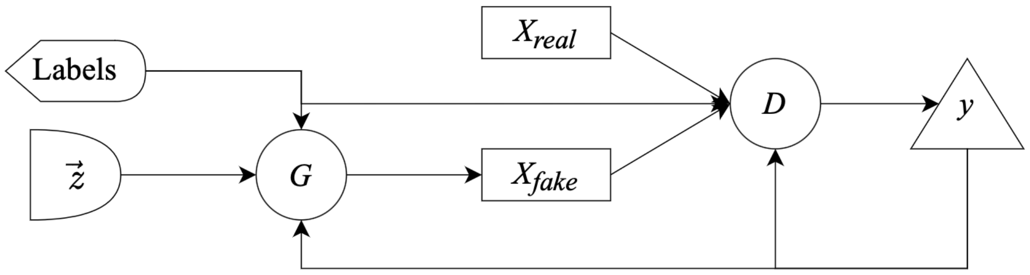


Figure 3.3: A diagram of a CGAN. The CGAN is similar to a vanilla GAN (see Figure 3.1), but the generator, G, and the discriminator, D, are conditioned on class labels.

Despite the importance of the CGAN, with its clear advantage of being able to draw a sample from a user-selected class, back in 2014 the generation of synthetic images had a lot of room for improvement. As such, a growing number of GAN architectures focused on image generation were proposed in the following years.

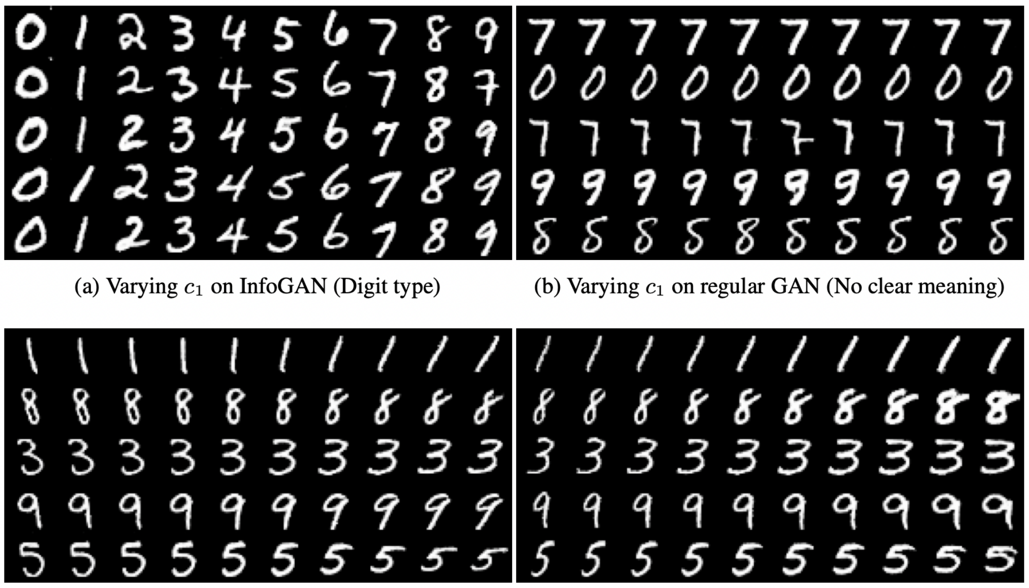
Deep Convolutional Generative Adversarial Network (DCGAN) is a GAN architecture that combines convolutional layers[4], which are commonly used in computer vision tasks, with GANs. Radford *et al.*, in [65], have brought together the success of Convolutional Neural Networks (CNNs) in supervised learning tasks with the then emerging GANs. Nowadays, the use of convolutional layers in GANs for image generation is quite common, but at that time (2016) this was not the case. Therefore, the use of convolutional layers in the GAN structure is still a powerful tool for handling image data.

Thus, the DCGAN was able to enhance the generated images by using convolutional layers. However, the features in the latent space had no semantic meaning. That is, it was not possible to change the values of a feature in latent space and predict what that change would do to the image, (*e.g.*, rotation, widening).

Information Maximizing Generative Adversarial Network (InfoGAN), is a GAN extension proposed by Chen *et al.* in [20], that attempts to learn disentangled information. That is, to give semantic meaning to features in the latent space (see Figure 3.4). InfoGAN can successfully recognize writing styles from handwritten digits in the MNIST dataset, detect hairstyles or eyeglasses in the CelebA dataset, or even background digits from the central digit

[4]A convolutional layer is a layer that uses a convolution operation. A convolution, in terms of computer vision tasks, consists of a filter (represented by a matrix) that slides through the image pixels (also represented by a matrix) and performs matrix multiplication. This is useful in computer vision tasks because applying different filters to an image (by means of a convolution) can help, for example, detect edges, blur the image, or even remove noise.

in the SVHN dataset.



(a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation) (d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure 3.4: The semantic meaning InfoGAN adds to the latent variables in the MNIST dataset. In (**a**), varying the latent variable c_1 leads to a digit change (from 0 to 9), while in (**b**), a regular GAN does not add meaning to its latent variables. In (**c**), the variation of c_2 leads to the rotation of digits. Finally, in (**d**), variation c_3 controls the width of the digits. Image taken from [20].

The GAN architectures presented so far can be quite time-consuming and use a high amount of computing resources to train. Given a large number of hyperparameters and a large number of training samples, the training process could be prohibitively expensive due to the training time and resources required.

Coupled Generative Adversarial Networks (CoGAN), proposed in [49] by Ming-Yu Liu and Oncel Tuzel, use a pair of GANs instead of only one GAN. The CoGAN was used to learn the joint distribution of multi-domain images, which was achieved by the weight-sharing constraint between the two GANs. In addition, sharing weights requires fewer parameters than two individual GANs, which, in turn, results in less memory consumption, less computational power, and fewer resources.

The focus on image generation continued, and in 2016, the AC-GAN and the StackGAN architectures were introduced to provide improvements in synthetic image generation.

Auxiliary Classifier Generative Adversarial Network (AC-GAN) [58], is a GAN extension proposed by Odena *et al.* that modifies the generator to be class dependent (it takes class labels into account) and adds an auxiliary model to the discriminator whose purpose is to reconstruct the class label. The results in [58] show that such an architecture can generate globally coherent samples that are comparable, in diversity, to those of the ImageNet dataset (see Figure 3.5).

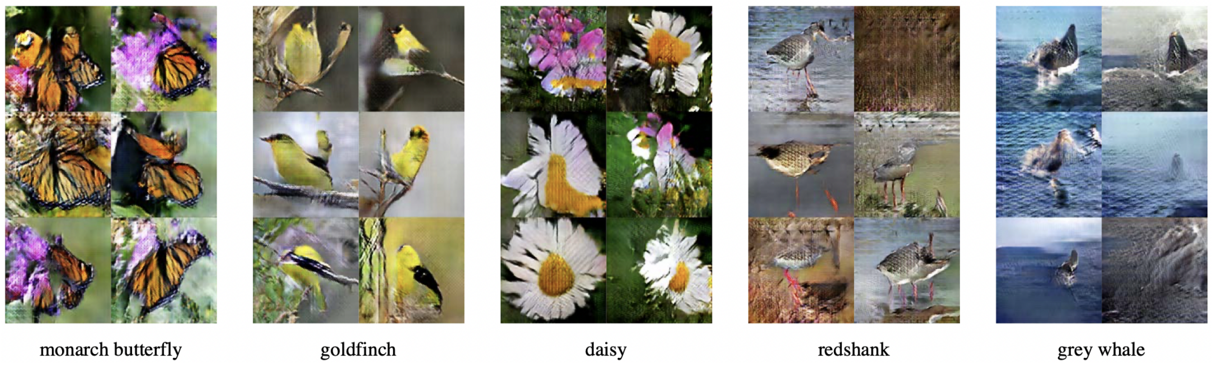


Figure 3.5: Images of five distinct classes generated by the AC-GAN. Nowadays, the detail in the images is far superior to the one provided by the AC-GAN. Image taken from [58].

Stacked Generative Adversarial Network (StackGAN), proposed in [90] by Zhang *et al.*, is another extension of GANs that can generate images from text descriptions. This generation of photorealistic images is decomposed into two parts. First, the STAGE-I GAN sketches a primitive shape and colors based on the input text. Next, the Stage-II GAN uses the same text description as the STAGE-I GAN and its output as input and generates high-resolution images by refining the output images by STAGE-I GAN. Their work has led to significant improvements in image generation.

Despite improving the quality of the generated images, adding semantic meaning to the latent features, and reducing memory consumption and training time, the training itself was still difficult due to mode collapse and uninformative loss metrics.

Wasserstein Generative Adversarial Networks (WGAN), is an alternative to traditional GAN training. The WGAN proposed by Arjovsky *et al.* in [9] is a GAN extension that modifies the training phase such that the discriminator, called the critic, is updated more often than the generator at each iteration i, where i is defined by the user. This change to GAN training avoids mode collapse and provides a meaningful loss metric that correlates with the generator's convergence and sample quality.

Returning to image generation, an interesting idea is to transfer an image from one area to another. For example, let us take a landscape image and "merge it" with an image of a Monet painting so that the landscape image has the style of a Monet painting.

Cycle-Consistent Generative Adversarial Network (CycleGAN), is a GAN extension for image-to-image translation without paired data. Zhu *et al.* proposed, in [97], an approach to translate an image from a domain X to a domain Y when no paired images are available. The CycleGAN consists of two generators, G and F, and two discriminators, D_X and D_Y. G maps an image from X to Y, and D_Y tries to determine whether it is from the original dataset or synthesized. Similarly, F maps an image from Y to X, and D_X determines whether it is real or generated by F. In addition to the four networks, the cycle consistency loss metric was also introduced to ensure that translating an image from X to Y and then from Y to X yields a very similar image to the original one. Figure 3.6 shows the image-to-image translation capabilities of CycleGAN.

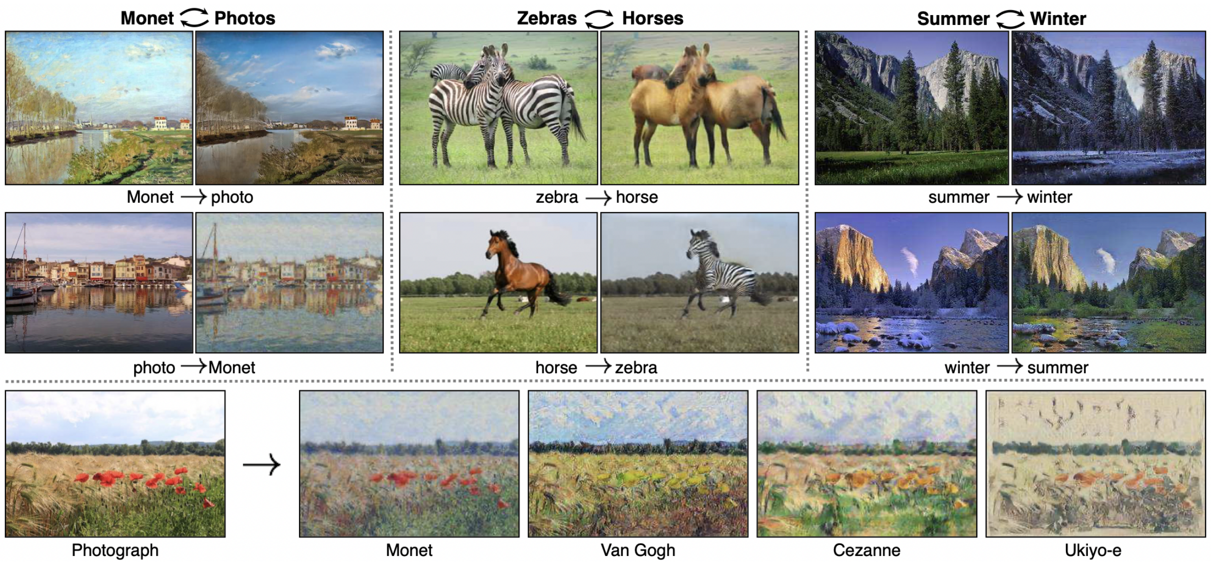


Figure 3.6: Given any two image collections, the CycleGAN learns to automatically "translate" an image from one domain into the other and vice versa. Example application (bottom): using a collection of paintings of famous artists, the CycleGAN renders a user's photograph in their style. Image taken from [97].

To date, GAN architectures have focused on image generation and translation, training stabilization, and time or have been tied to class labels. Nonetheless, there is an interesting application of GANs to music generation.

Multi-track sequential GAN (MuseGAN), proposed by Dong *et al.* in [25] is a GAN architecture for music generation. This is quite different from generating images or videos since music has a temporal dimension, is usually composed of multiple instruments, and musical notes are often grouped into chords. Although the music generated is not as good as that produced by professional musicians, the results were quite promising, and the MuseGAN model had some interesting properties.

In late 2017 and throughout 2018, the quality of image-generated data improved greatly with the introduction of ProGAN, SAGAN, and BigGAN architectures.

Progressive growing of Generative Adversarial Networks (ProGAN), is a technique that helps stabilize GAN training by progressively increasing the resolution of generated images. Proposed in [43] by Karras *et al.*, the ProGAN accelerates and stabilizes training by, first, constructing a generator and a discriminator that produce images with few pixels. Then, layers corresponding to higher resolutions are added in the training process, allowing the creation of high-quality images. Figure 3.7 shows images generated with the ProGAN.

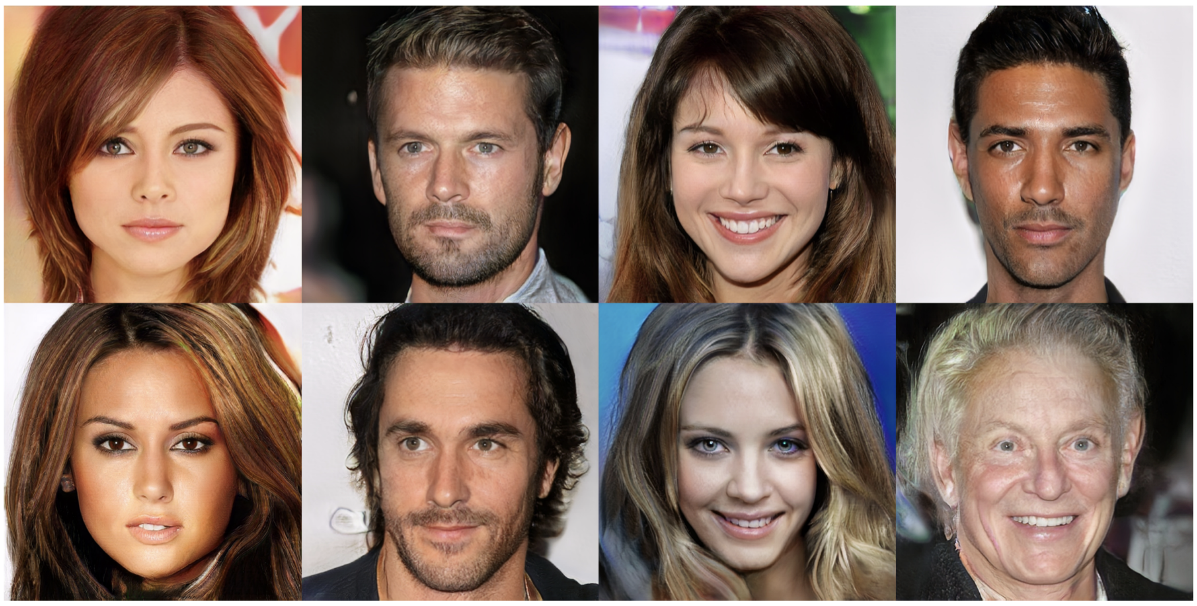


Figure 3.7: Images generated using ProGAN. Notice the level of detail when compared to the ones generated from AC-GAN (Figure 3.5). Image taken from [43].

Self-Attention Generative Adversarial Networks (SAGAN), improves on previous GAN structures by maintaining long-range relationships within an image rather than just local points [91]. Zhang *et al.* have found that using spectral normalization improves the training dynamics of the generator. In addition, the discriminator can assess whether highly detailed features in distant image regions match each other. When this architecture was proposed, the authors were able to improve both the Inception Score [70] and the Fréchet Inception Distance [40] (two widely used metrics to evaluate synthetic image data) on the ImageNet dataset.

Big Generative Adversarial Network (BigGAN), proposed by Brock *et al.* [14], is a type of GAN architecture that upscales existing GAN models and produces high-quality images (see Figure 3.8). BigGAN has also demonstrated how to train GANs at a large scale by introducing techniques that detect training instability. At the time of BigGAN's introduction, its performance was significantly better than that of other state-of-the-art structures.

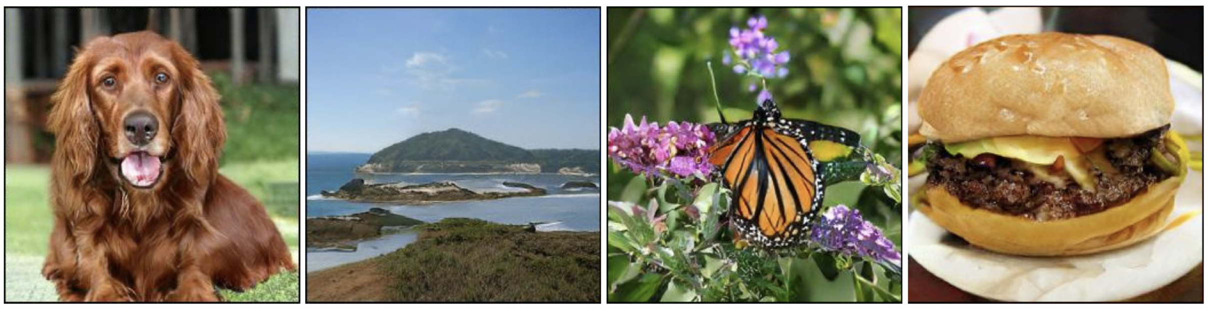


Figure 3.8: Class-conditional samples generated by BigGAN. Image taken from [14].

As seen previously, the image quality has improved considerably (compare Figure 3.5 with Figures 3.7 and 3.8, for example). However, there were still some limitations in the images generated. Although the GAN architectures provided extremely realistic images, it was still difficult to understand various aspects of the image synthesis process [44].

Style-based Generative Adversarial Networks (StyleGAN), proposed by Karras *et al.* in [44], explores an alternative generator architecture based on style transfer. The focus is not on generating more realistic images but on having better control over the generated image. This new architecture is able to learn to separate high-level features and stochastic variation. In fact, the new generator improves the quality metrics over the state-of-the-art, untangles the latent variables better, and has better interpolation properties.

Two other different ideas than those shown so far, but also very interesting, were proposed in 2019. The first is about turning a user's sketch into a realistic image. The second is about automatically completing an incomplete image in a plausible way.

GauGAN [62], a model proposed by NVIDIA Research that allows users to sketch an abstract scene and then turn it into a detailed image. Users can also manipulate the scene and label each element. This is achieved through the use of a spatially-adaptive normalization layer whose purpose is to aid in the generation of photorealistic images when a semantic layout is given as input.

Pluralistic Image Inpainting GAN (PiiGAN), proposed by Weiwei Cai and Zhanguo Wei in [16], attempts to fill in large missing areas in an image. Unlike other Deep Learning (DL) methods that try to achieve a single optimal result, PiiGAN has a new style extractor that is able to extract the style features from the original images. As shown in [16], PiiGAN can produce images of better quality and greater variety than other state-of-the-art architectures that match the context semantics of the original image. Figure 3.9 shows the capabilities of PiiGAN.

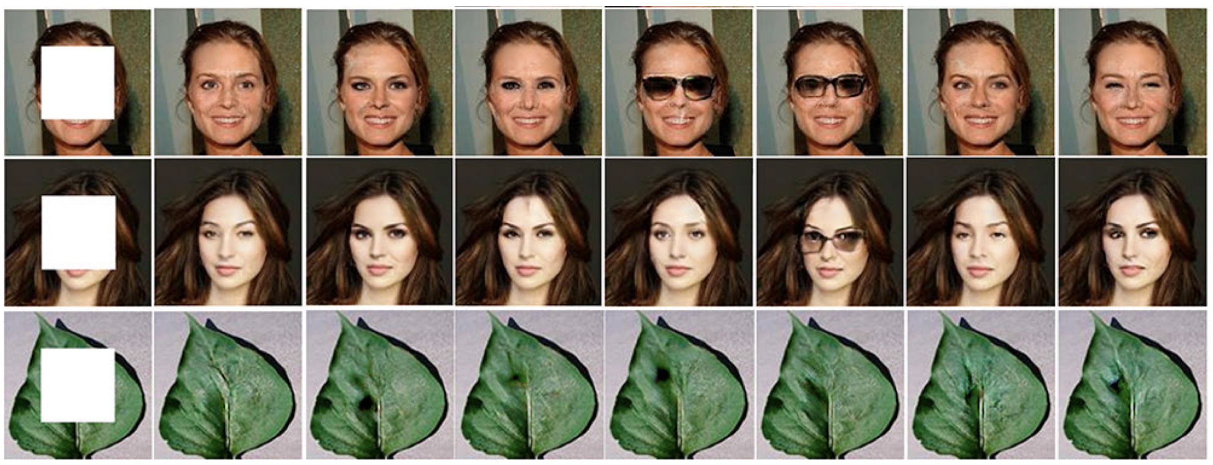


Figure 3.9: Examples of inpainting results produced by PiiGAN on two faces and a leaf. On the left column is the input image (with the center pixels removed). The images in the remaining columns are outputs of the PiiGAN. Image taken from [16].

A more recent architecture, introduced in 2021, is the Multy-StyleGAN, which highlights the capabilities of GANs in various image domains – in this case, biology.

Multi-StyleGAN, proposed by Prangemeier *et al.* in [64], is a novel GAN architecture used to study the dynamic processes of life at the level of single cells. Since acquiring images to study such processes is costly and complex, the Multi-StyleGAN is a descriptive approach that simulates microscopic images of living cells. As shown by the authors, the proposed architecture is capable of capturing the underlying biophysical factors and temporal dependencies.

As shown in the previous paragraphs, the major breakthroughs of GANs are focused on imaging generation. Despite their enormous success in this area, GANs can be used in other areas as well since there are no restrictions on whether the dataset must be an image, a video, music, or an ordinary tabular dataset. Nonetheless, different types of architectures must be considered depending on the task. Image data does not have the same characteristics as music or tabular data, so different types of layers, activation functions, or training procedures must be selected accordingly. That being said, there are some best practices that can be used depending on the data at hand, but the architecture of a GAN currently seems to be as much an art as a science. In the next subsection, we take a closer look at three GAN structures used to generate tabular data.

3.4 GANs for Tabular Data

As seen in Section 3.3, GANs are widely and successfully used for image generation tasks. However, many datasets have a tabular format, and the most popular GAN architectures cannot be used in such a setting because tabular data has unique properties.

First, continuous and categorical features are present in most tabular datasets. Since image data consists solely of numerical features (the pixels), GANs used for image generation tasks cannot accommodate the different types of variables. Second, non-Gaussian and multimodal distributions are quite common in tabular datasets. Numerical features in tabular data may have multiple modes and follow a non-Gaussian distribution, which must be considered when generating synthetic data. Third, highly imbalanced categorical variables are common. This can lead to severe mode-collapse and insufficient training for the minority classes. Finally, it is easier for a trivial discriminator to distinguish between real and fake data when it learns from sparse one-hot-encoded vectors since it takes into account the sparsity of the distribution rather than checking the overall authenticity of the sample.

In the following paragraphs, we detail three important GAN architectures used to overcome the above problems. The TGAN architecture was introduced in 2018, followed by the CTGAN architecture in 2019, which is an evolution of the TGAN architecture and was proposed by the same authors. This was followed in 2021 by the TabFairGAN, which was intended to dethrone the two aforementioned GANs in terms of the quality of synthetic tabular data generation. We believe the detailed explanations that follow can shed some light on a topic that is as not as well disseminated in the literature, as far as we are aware – the use of GANs to generate tabular data rather than image data.

3.4.1 TGAN

TGAN was proposed in 2018 by Lei Xu and Kalyan Veeramachaneni [88] as a GAN architecture for synthesizing tabular data. Given a dataset, unbal, which is already split into train set, D_{train}, and test set, D_{test}, the aim of the TGAN is twofold: given a Machine Learning (ML) model, its accuracy on D_{test} when trained on the D_{train} should be similar to its accuracy, also on D_{test}, but when trained using D_{synth}, which is the synthetic data (machine learning efficacy); the mutual information between each pair of columns in D and D_{synth} should be similar.

To achieve these goals, first, it is important to transform the data. A GAN usually consists of two neural networks, so it is crucial to properly represent the data before feeding it as input. This problem is addressed by applying mode-specific normalization for numerical variables and smoothing for categorical variables.

Mode-specific normalization is used to handle non-Gaussian and multimodal distributions. It fits a Gaussian Mixture Model (GMM), which models a distribution as a weighted sum of Gaussian distributions to each numerical variable and calculates the probability that a sample from a numerical column comes from each of the Gaussian distributions. These probabilities are then used to encode the values of the rows corresponding to the numerical features. More formally, let $\{N_1, N_2, \ldots, N_p\}$ represent the numerical columns of a tabular dataset D. A GMM with m components is fitted to each numerical variable, N_i. The means and standard deviations of the m Gaussian distribution are represented by $\mu_i^{(1)}, \mu_i^{(2)}, \ldots, \mu_i^{(m)}$ and $\sigma_i^{(1)}, \sigma_i^{(2)}, \ldots, \sigma_i^{(m)}$, respectively. The probability of $x_{i,j}$ (the value at row i and column j) coming from each of the m Gaussian distributions is given by a vector $u_{i,j}^{(1)}, u_{i,j}^{(2)}, \ldots, u_{i,j}^{(m)}$. Finally, $x_{i,j}$ is normalized as $v_{i,j} = \frac{x_{i,j} - \mu_i^{(k)}}{2\sigma_i^{(k)}}$, where $k = argmax_k u_{i,j}^{(k)}$ and $v_{i,j}$ is clipped to $[-0.99, 0.99]$, and u_i, v_i are used to encode x_i.

Smoothing of the categorical variables is achieved by representing them as one-hot-encoded vectors, adding noise to each dimension (drawn from a uniform distribution), and renormalizing the vector. After applying mode-specific normalization to the numerical columns and smoothing the categorical ones, the data are ready to be fed into the TGAN. The generator is a Long-Short Term Memory (LSTM) network that generates the numeric variables in two steps (in the first step, v_i is generated, and u_i is generated in the second step) and the categorical variables in one step. A fully connected neural network is used as the discriminator. A diagram of a TGAN is shown in Figure 3.10.

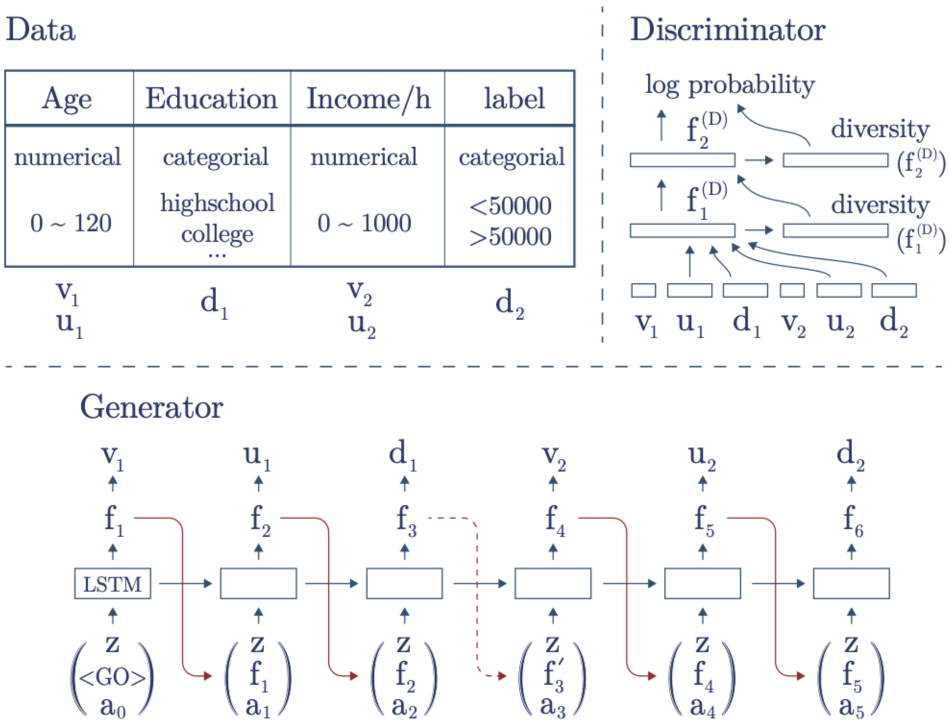


Figure 3.10: Diagram of a TGAN used in a toy example with 2 continuous and 2 discrete variables. Image taken from [88].

The TGAN was evaluated, in [88], with respect to machine learning efficacy and the preservation of correlation (the two aforementioned aims of the TGAN) and compared with other data synthesis models. Regarding machine learning efficacy, five models were evaluated in terms of accuracy and Macro-F1, namely, Decision Trees, Linear Support Vector Machines, Random Forests, AdaBoost, and Multi-Layer Perceptrons, on three different datasets. It was found that while the ML models generally performed better when trained on the real dataset, the average performance difference between the real and synthetic data was 5.7%. This suggests that the TGAN performs quite well (The authors compared the TGAN with a Gaussian Copula (GC) and a Bayesian Network (BN-Co), which showed a drop in performance of 24.9% and 43.3%, respectively). Moreover, the TGAN was able to maintain the ranking of the ML models. As for the preservation of correlation between any two pairs of variables, the TGAN was able to successfully capture this correlation.

3.4.2 CTGAN

The CTGAN, also proposed by Lei Xu and Kaylan Veeramachaneni *et al.* [89] in 2019, is an improvement over TGAN. The objectives of CTGAN are almost the same as those of TGAN. The difference is that CTGAN is more ambitious, and instead of just preserving the correlation between any pair of columns in the synthetic data, it aims to preserve the joint distribution of all columns.

As for the transformations of the input data, they are similar to those presented for the TGAN model. To transform the numerical columns, a variational Gaussian mixture model (VGM) is used instead of a GMM. The difference is that the VGM estimates the number of modes for each numerical column, unlike in the TGAN, where the number of modes is predefined and is the same for each numerical column. In addition, the continuous values are represented as a one-hot vector indicating the mode and a scalar indicating the value within the mode [5]. The categorical features are only one-hot-encoded without adding noise.

To allow the CTGAN to deal with imbalanced discrete columns, the authors used a conditional generator that can generate synthetic rows that depend on any of the discrete columns. Furthermore, a technique called training by sampling was proposed, allowing the CTGAN to uniformly examine all possible discrete values.

To integrate the conditional generator in the GAN architecture, it is necessary to properly prepare the input. This is accomplished by using a conditional vector, which specifies that a given categorical column must be equal to a certain value (from the set of the possible values for that particular column). Furthermore, the generator loss is modified so that it learns to map the conditional vector into the one-hot-encoded values. The conditional vector consists of a simple transformation to the one-hot-encoded vectors. Supposing that a dataset with 3 discrete columns, $D_1 = \{0, 1, 2\}, D_2 = \{0, 1\}, D_3 = \{0, 1, 2\}$, is given, and the condition that is indicated is $D_2 = 1$, the conditional vector would be

$$\vec{cond} = (\underbrace{0, 0, 0}_{D_1}, \underbrace{0, 1}_{D_2}, \underbrace{0, 0, 0}_{D_3})$$

Where the first three entries correspond to the one-hot-representation of D_1, the fourth and fifth entries correspond to the one-hot representation of D_2, and the last three entries correspond to the one-hot representation of D_3. The conditional generator is then forced to map the conditional vector into the one-hot-encoded ones by adding the cross entropy to its loss function.

Training by sampling is a technique that ensures that the conditional vector is properly sampled so that the CTGAN can uniformly examine all possible values in discrete columns. This is performed by randomly selecting a discrete column, constructing the probability mass function over the possible values for the selected column (the probability mass of each value is the logarithm of its frequency), and only then computing the conditional vector. A diagram of a CTGAN is shown in Figure 3.11 (the conditional generator and the discriminator are both fully-connected networks).

[5] *e.g.*, if the VGM has estimated three modes and a given value $x_{i,j}$ has a greater probability of coming from mode 2, then the one-hot-encoded vector would be $\vec{\beta} = (0, 1, 0)$ and the value within the mode would be given by $a_{i,j} = \frac{x_{i,j} - \mu_2}{4\sigma_2}$, where μ_2 is the mean of the Gaussian distribution corresponding to the second mode, and σ_2 its standard deviation.

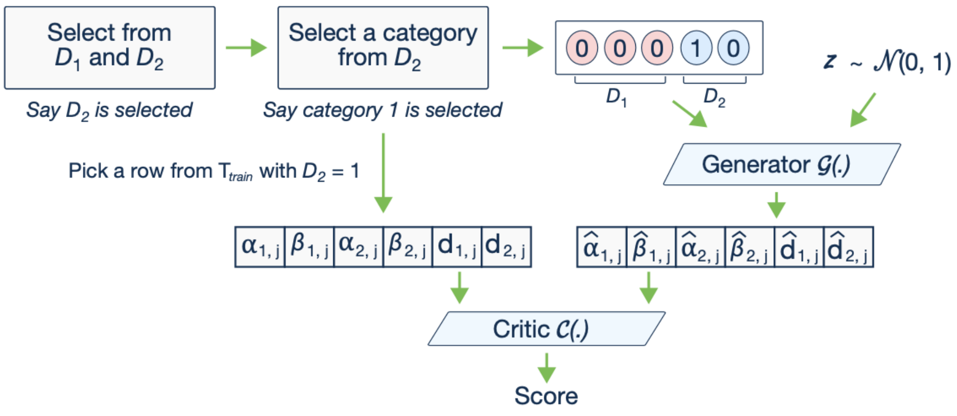

Select from D_1 and D_2
Say D_2 is selected
Select a category from D_2
Say category 1 is selected
0 0 0 1 0
D_1
D_2
$z \sim \mathcal{N}(0, 1)$
Pick a row from T_{train} with $D_2 = 1$
Generator $\mathcal{G}(.)$
$\alpha_{1,j}$ $\beta_{1,j}$ $\alpha_{2,j}$ $\beta_{2,j}$ $d_{1,j}$ $d_{2,j}$
$\hat{\alpha}_{1,j}$ $\hat{\beta}_{1,j}$ $\hat{\alpha}_{2,j}$ $\hat{\beta}_{2,j}$ $\hat{d}_{1,j}$ $\hat{d}_{2,j}$
Critic $\mathcal{C}(.)$
Score

Figure 3.11: Diagram of a CTGAN. Image taken from [89].

To evaluate the CTGAN, the authors in [89] have used seven simulated datasets and eight real datasets. In the simulated datasets, the likelihood fitness metric was computed to evaluate performance, which is possible since the distribution of the data is known. In what concerns the real datasets, the machine learning efficacy was used to evaluate performance (it is not possible to compute the likelihood fitness metric in real datasets because the distribution of the data is unknown). The CTGAN was also compared with other generative models, namely CLBN [23], PrivBN [92], MedGAN [21], VeeGAN [78], and TableGAN [61]. It was found that in real datasets, the CTGAN outperformed all other models in terms of machine learning efficacy. In simulated datasets, the CTGAN performed quite well in terms of the likelihood fitness metric, although it was not able to outperform all other models.

Finally, an ablation study was conducted with the goal of evaluating the utility of mode-specific normalization, conditional generator, and training by sampling. The results showed that if the mode-specific normalization was replaced by either a GMM with five modes (GMM5), a GMM10, or a min-max normalization, the losses in performance (regarding machine learning efficacy) in the real datasets would be of -4.1%, -8.6%, and -25.7%, respectively. In what concerns the training by sampling, if removed, the performance would decrease by 17.8%. If the conditional generator was removed, the performance would drop by 36.5%. Therefore, the techniques introduced in CTGAN, namely, mode-specific normalization, training by sampling, and the conditional generator, are very important for generating high-quality tabular data.

3.4.3 TabFairGAN

TabFairGAN, proposed in 2021 by Amirarsalan Rajabi and Ozlem Ozmen Garibay [66], is a WGAN with a gradient penalty. As with TGAN and CTGAN, it is crucial to represent the data correctly before entering it as input to the TabFairGAN. Thus, Rajabi and Garibay used one-hot-encoding to represent the categorical features. A quantile transformation was used for the numerical features:

$$c_i' = \Phi^{-1}(F(c_i))$$

where c_i is the i^{th} numerical feature, F is the cumulative distribution function (CDF) of the feature c_i, and Φ is the CDF of a uniform distribution.

In what concerns the network structure, the generator is formally described as:

$$\begin{cases} h_0 = z \\ h_1 = ReLU(FC_{l_w \to l_w}(h_0)) \\ h_2 = ReLU(FC_{l_w \to N_c}(h_1)) \oplus gumbel_{0.2}(FC_{l_w \to l_1}(h_1)) \oplus \\ gumbel_{0.2}(FC_{l_w \to l_2}(h_1)) \oplus \ldots \oplus gumbel_{0.2}(FC_{l_w \to N_d}(h_1)) \end{cases}$$

where z is a latent variable drawn from a standard multivariate normal distribution, $ReLU$ is the rectified linear unit activation function, $FC_{a \to b}$ denotes a fully connected layer with input size a and output size b, l_w is the dimension of an input sample, N_c is the number of numerical columns, N_d is the number of categorical columns, l_i is the dimension of the one-hot-encoded vector of the i^{th} categorical column, $\oplus$ denotes the concatenation of vectors, and $gumbel_\tau$ is the Gumbel softmax with parameter τ (a continuous distribution that approximates samples from a categorical distribution and uses backpropagation).

In what concerns the critic (discriminator), its architecture can be formally described as follows:

$$\begin{cases} h_0 = X \\ h_1 = LeakyReLU_{0.01}(FC_{l_w \to l_w}(h_0)) \\ h_2 = LeakyReLU_{0.01}(FC_{l_w \to l_w}(h_1)) \end{cases}$$

Here X denotes the output of the generator or the transformed real data, and $LeakyReLU_\tau$ represents the leaky rectified linear unit activation function with slope τ. Figure 3.12 shows a diagram of the TabFairGAN. An initial fully connected layer (with $ReLU$ activation) constitutes the generator, followed by a second layer that uses $ReLU$ for numerical attributes and Gumbel softmax for one-hot-encoding of the categorical features. In the last layer, all the attributes are concatenated, producing the final generated data. The critic is constituted by fully connected layers with the $LeakyReLU$ activation function.

TabFairGAN was evaluated in terms of machine learning efficacy (the F1-score and accuracy metrics were used) using three ML models, namely, decision trees, logistic regression, and multi-layer perception in the UCI Adult Income Dataset. The results were compared with two other state-of-the-art models, the TGAN and the CTGAN. TabFairGAN was found to perform better than TGAN and CTGAN on all ML models and metrics used, except MLP, where CTGAN performed better than TabFairGAN in terms of accuracy (but not in the F1-score). Hence, the TabFairGAN is quite effective in generating data similar to the real tabular data.

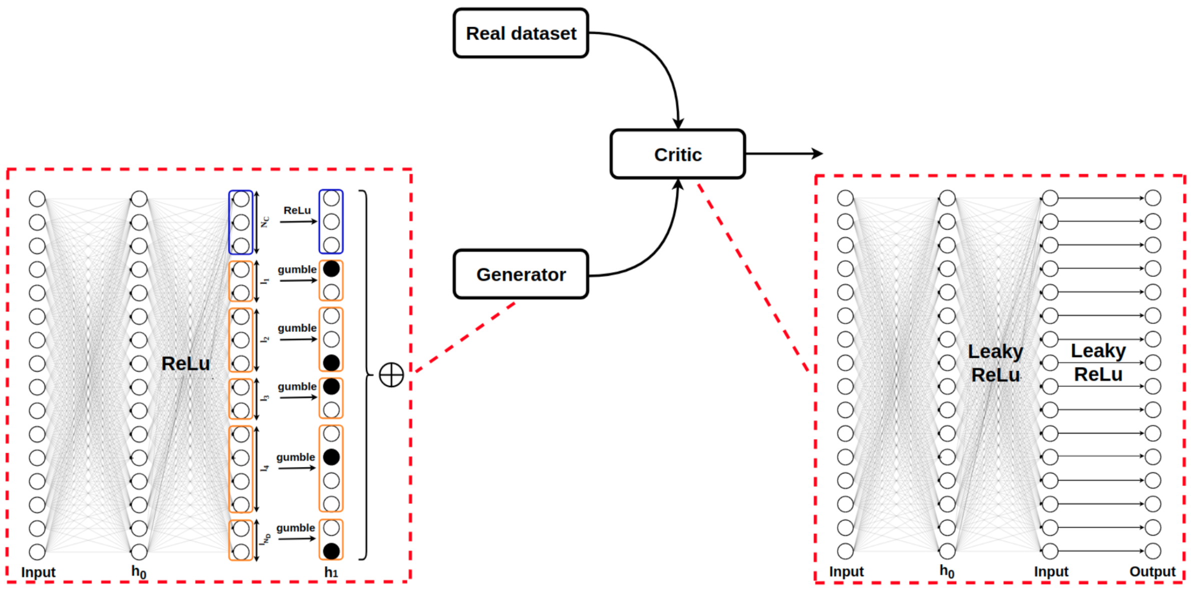


Figure 3.12: TabFairGAN architecture. Image taken from [66].

Having shown the several synthetic data generation methods in Chapter 2 and then funneling down to GANs in the current chapter, a next logical step is to present how the quality of synthetic data can be evaluated. This is, indeed, quite important as bad quality synthetic data can be useless or even jeopardize the performance of ML models. The following chapter (Chapter 4) guides the reader through the most common synthetic data evaluation methods.

Chapter 4

Synthetic Data Quality Evaluation

Evaluating the quality of the generated samples is critical to assessing the quality of the method used to generate synthetic data. There is a huge number of evaluation techniques, so it is tedious and almost impossible to explore and describe them all. Moreover, many of these evaluation techniques are intended for specific types of data or for very specific domains. We focus our attention only on tabular data, disregarding the domain to which the data belongs, as we aim to use general evaluation methods, not restricting ourselves to a specific domain. Therefore, this chapter focuses on evaluation methods for tabular data.

4.1 Synthetic Data Evaluation Methods

The simplest way to evaluate the quality of synthetic data is to compare their basic statistics, (*e.g.*, mean, median, standard deviation) with those of the real data. If the values are similar, it is likely that the synthetic data are similar to the real data. However, this can be misleading, as statistician Francis Anscombe showed in 1973 [8]. The Anscombe quartet includes four datasets that are nearly identical in terms of basic descriptive statistics but whose distributions are very different.

Anscombe constructed his quartet to demonstrate the importance of plotting the data when analyzing it (see figure 4.1). Back in 1973, it may have been challenging to create plots with data, in part because of scarce and expensive computing resources. Today, however, it is quite easy, with hundreds of graphics libraries available for various programming languages. Thus, another method to evaluate the quality of synthetic data is to use graphical representations, (*e.g.*, box plots, histograms, violin plots).

Comparing the graphs of the synthetic data with the graphs of the real data provides a visual assessment of the generated data, which can also be supplemented by descriptive statistics. The Q-Q plot is a probability plot that can be particularly useful for making comparisons between two data distributions, as it plots their quantiles against each other and can, thus, evaluate the similarity between the distributions (see figure 4.2). Given the vast amounts of data available

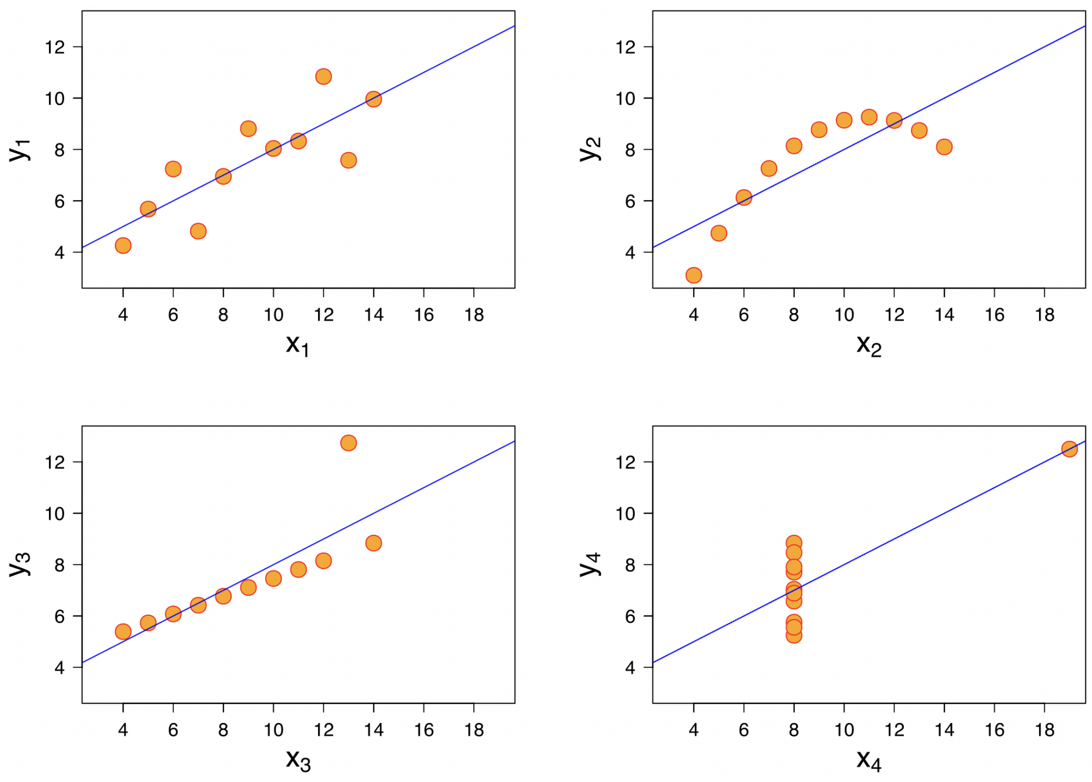
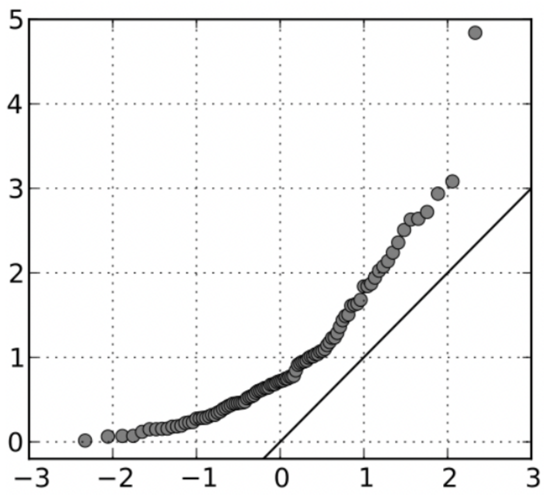


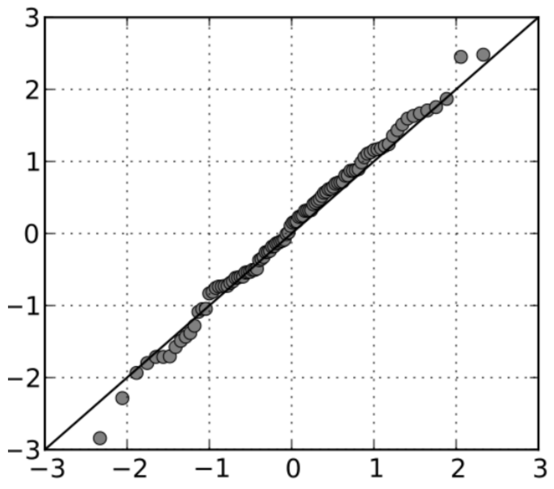
Figure 4.1: Ascombe's quartet. All four sets are identical when examined using simple summary statistics but vary considerably when plotted.

today, with datasets containing hundreds or even thousands of variables, it can be prohibitively expensive to visually represent and analyze all of the data, so other approaches to evaluate synthetic data are required.

Machine learning efficacy is another technique for evaluating synthetic data. Since many of the uses of synthetic data are to increase the performance of Machine Learning (ML) models, machine learning efficacy is used to evaluate the quality of synthetic data with respect to the performance of ML models. It consists of, given a dataset, D, already split into a train set, D_{train}, and test set, D_{test}, comparing the performance of ML models, (*e.g.*, logistic regression, decision trees, artificial neural networks, etc) when trained in D_{train}, and on D_{synth} (the synthetic data), and evaluated in D_{test} (see Figure 4.3). If the performance, (*e.g.*, in terms of accuracy, recall, precision, F1-score) of the models trained using D_{train} is similar to those trained using D_{synth}, then the synthetic data is likely to follow the underlying data distribution. In [88, 89], this method was used to evaluate the performance of the TGAN and CTGAN architectures, respectively. Furthermore, in [96], this technique was used to evaluate the forecast of emerging technologies.



(a) An example of a Q-Q plot in the case where the two distributions are not similar.



(b) An example of a Q-Q plot in the case where the two distributions are very similar.

Figure 4.2: The Q–Q plot is used to compare the shapes of distributions, providing a graphical view of how similar the two distributions are. If the two distributions being compared are similar, the points in the Q–Q plot will approximately lie on the identity line $y = x$.

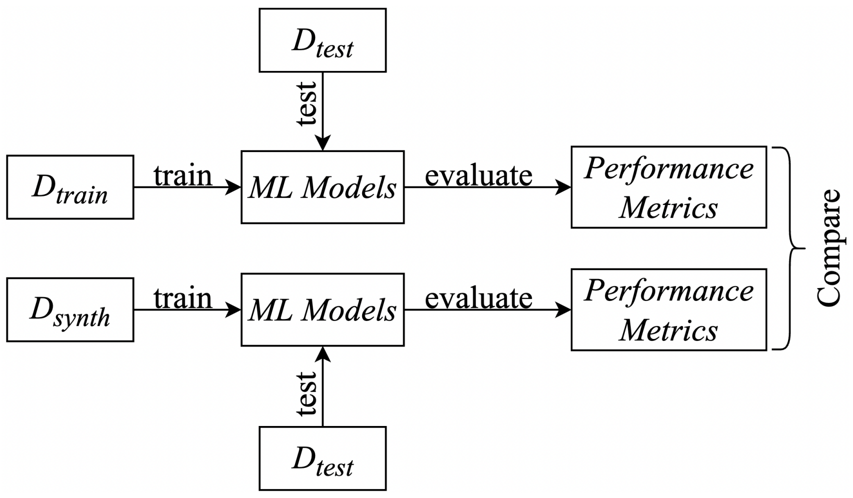


Figure 4.3: Machine learning efficacy diagram. The performance (accuracy, F1-score, etc.) of ML models (random forests, decision trees, etc.) in the test set, D_{test}, is compared when the models are trained on the real training data, D_{train}, and when they are trained using the synthetic data, D_{synth}.

In [73], Shmelkov *et al.* argue that the existing methods for evaluating synthetic samples are insufficient and must be adapted to the task at hand. They begin by addressing two commonly used metrics, namely the Inception Score [70] and the Fréchet Inception Distance [40][1]. Both

[1]Both the Fréchet Inception Distance and the Inception Score are metrics used to assess the quality of images created by a generative model. However, unlike the Inception Score, which evaluates only the distribution of generated images, the Fréchet Inception Distance compares the distribution of generated images with the

metrics are used for the evaluation of image-generated data and, thus, are not the focus of this work. Nonetheless, it is important to mention them at least, as they are widely used in the literature to evaluate synthetic image data.

After presenting these two metrics, the authors introduced their proposed metrics – GAN-train and GAN-test – which, although applied to image data, can also be applied to other types of data, such as tabular datasets. Moreover, despite both measures having "GAN" in their name, the synthetic samples do not need to be generated exclusively with a Generative Adversarial Network (GAN) but can also be generated with any other method. Therefore, we have slightly modified the definition of GAN-train and GAN-test given in [73] to make it more general by replacing the use of a GAN with any synthetic data generation method and the image data with any type of data (see Figure 4.4).

GAN-train. A classifier is trained with instances generated by a synthetic data generation method and its performance is evaluated against a test set consisting of real-world data. This measure provides a measure of how far apart the generated and true distributions are.

GAN-test. A classifier is trained on a real dataset and evaluated on the generated data. GAN-test provides a measure to evaluate whether the synthetic data generation method has overfitted (values significantly higher than the ones from validation accuracy) or underfitted (values significantly lower than the ones from validation accuracy) the data.

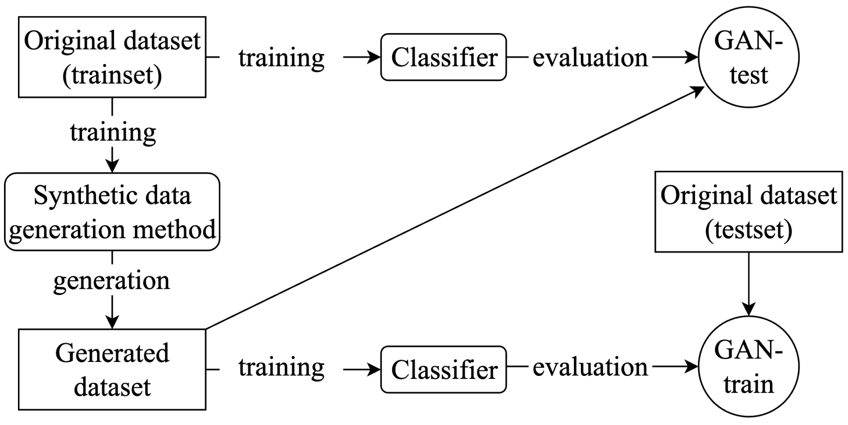


Figure 4.4: A diagram representing the GAN-train and GAN-test metrics. The GAN-train is a measure consisting of the accuracy of a classifier trained in the generated data and evaluated in the real data. GAN-test learns on the real data and is evaluated in the generated data. Image based on [73].

Another technique to evaluate the quality of synthetic data was proposed in [2]. The authors address the fact that most existing evaluation metrics for generative models are focused on image data and have introduced a domain- and model-independent metric. The metric is three-dimensional (α-Precision, β-Recall, Authenticity), and it evaluates the fidelity, diversity, and generalization of each generative model and is independent of the domain, (*e.g.*, images or tabular

distribution of real images.

data). Moreover, the three components correspond to interpretable probabilistic quantities, making it easier to detect a lack of synthetic data quality if such a problem occurs.

Fidelity. The α-Precision component measures the similarity between the generated and the real samples. Thus, values with high-fidelity correspond to realistic samples, *i.e.*, samples that resemble those from the original dataset.

Diversity. It is not enough to have samples that resemble those from the original dataset. High-quality synthetic data must also have some diversity. The β-Recall component evaluates how diverse the generated samples are. That is, whether the generated data is diverse enough to cover the existing variability in the real data.

Generalization. Last but not least, the generated samples mustn't be copies of the original data. High fidelity and diversity values do not guarantee that the synthetic samples are not just copies of the original dataset. Therefore, the authenticity component is a measure of how well the model can generalize and, therefore, not overfit the real data.

The first two components are computed by embedding the real and synthetic data in hyperspheres. That is, the original data, X_r, and the generated data, X_s, are mapped from the original domain, $\mathcal{X}$, to a hypersphere of radius r, $\mathcal{S}_r$. The third component is computed by evaluating the proximity of the real data to the generated data in the embedding space using a hypothesis test. Figure 4.5 shows a representation of the three metrics.

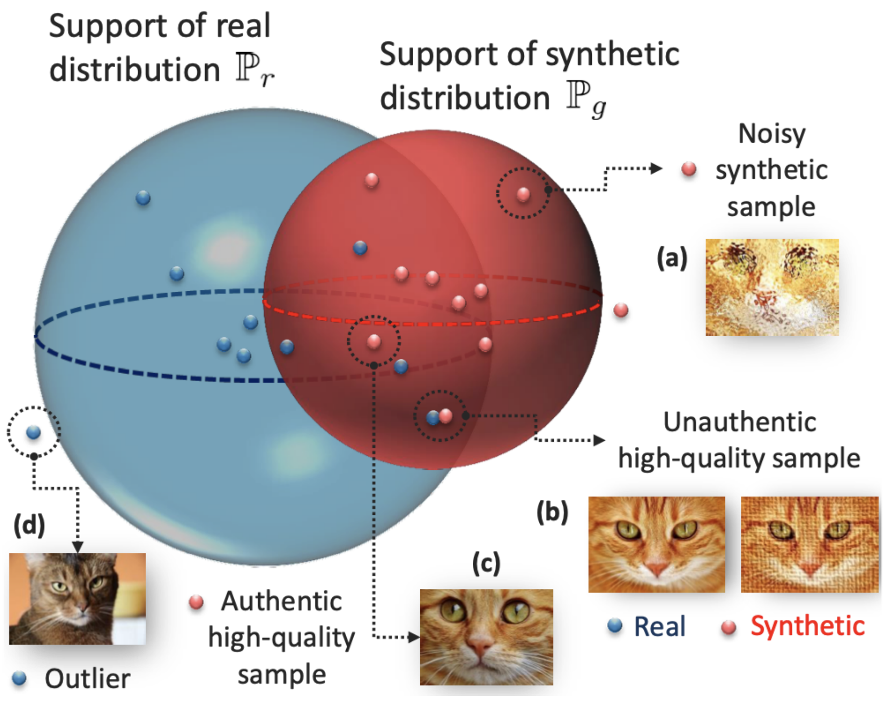

Support of real distribution $\mathbb{P}_r$
Support of synthetic distribution $\mathbb{P}_g$
Noisy synthetic sample
(a)
Unauthentic high-quality sample
(b)
(c)
(d)
Outlier
Authentic high-quality sample
Real
Synthetic

Figure 4.5: Representation of α-Precision, β-Recall, and Authenticity. The blue and red spheres correspond to the α- and β-supports of the real and the generated samples, respectively. Intuitively, these regions are "safe zones" where points that lie outside the spheres are outliers, and points inside the spheres are "ordinary" samples. (**a**) Generated samples that lie outside the blue sphere are unrealistic. (**b**) Synthetic samples that are very close to real instances are inauthentic because they are almost exact copies of the real data. (**c**) Synthetic data points inside the blue sphere and without real data points near them are considered high-quality samples. (**d**) A data point outside the sphere is considered an outlier. Image retrieved from [2].

Finally, an important aspect of this metric is that, unlike the other metrics, it provides the ability to evaluate each instance. Considering this, the authors of [2] have also proposed a model-checking framework where low-quality samples (low values in some or all components) are discarded. Therefore, the final generated dataset is a "curated" version consisting only of high-quality samples.

Last but not least, in [29], the authors have conceived a framework to evaluate the utility of synthetic data. They have used several utility metrics with concern to univariate, bivariate, and multimodal distributions, as well as the distinguishability of the dataset (how distinguishable is the synthetic data from the real one). Image 4.6 shows an illustration of the utility framework. As their approach is quite general, we have used their core ideas with some modifications to fit our use cases in chapter 5. As such, in the following paragraphs, we describe the approach so that the reader can understand the next chapter.

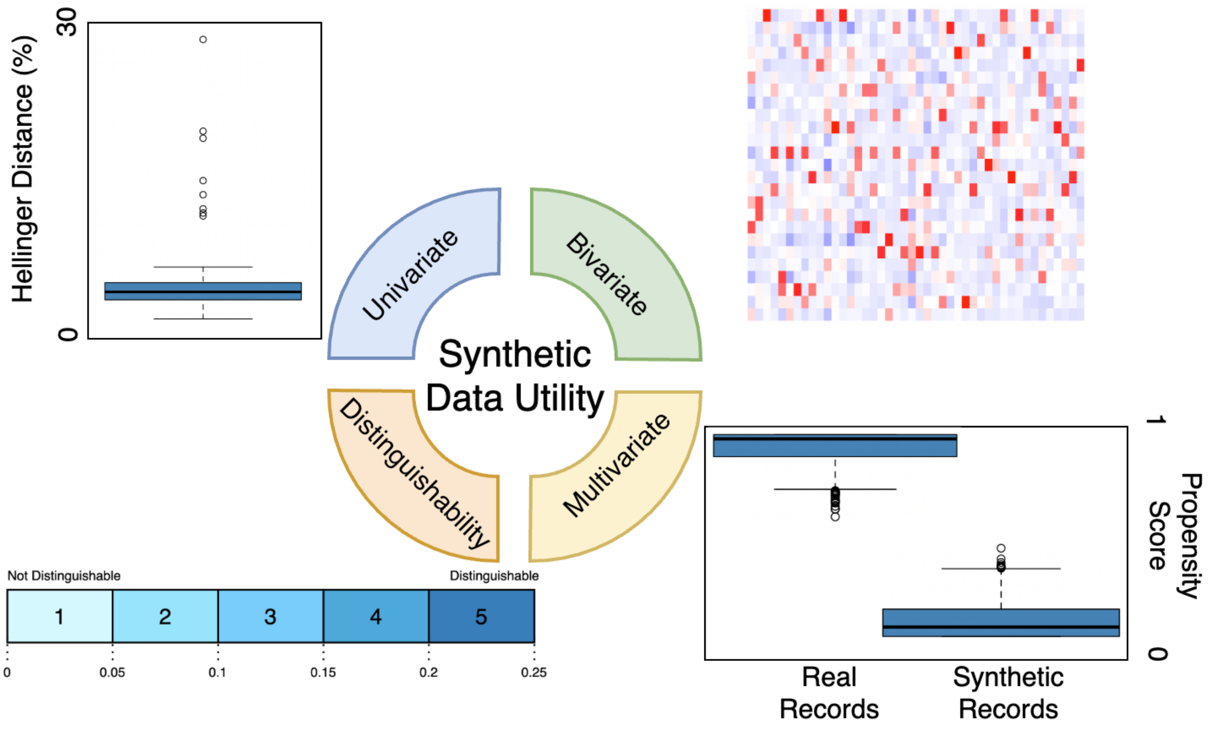


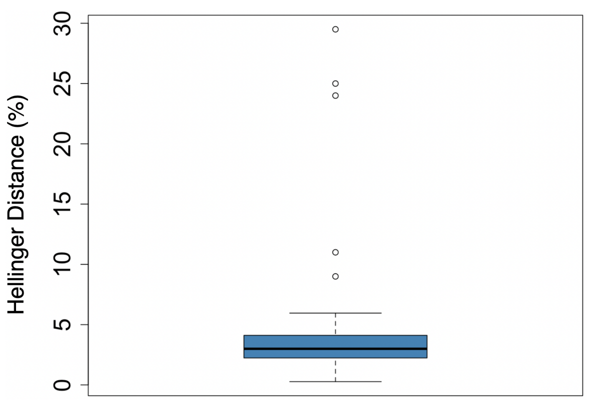
Figure 4.6: Illustration of the utility framework.

4.1.1 Univariate Analysis

Regarding the univariate analysis, it consists in comparing each feature in the synthetic data with the same feature in the original data. If the distributions are similar, but not exactly the same (this would indicate that the generated samples are just a copy of the original ones), then the feature was properly generated. This has an obvious problem, which is the lack of scalability: the more columns a dataset has, the more time it takes to evaluate every univariate distribution. As such, some sort of summary statistic is required to compare the real and synthetic features in a concise way.

For such purposes, the authors mention the Hellinger distance [39], which is a probabilistic measure between 0 and 1 used to measure the difference in the distributions (a value of 0 indicates no difference between distributions). An advantage of the Hellinger distance is that it deals with both continuous and categorical features. Also, when the dataset has many variables, the Hellinger distances can be represented in a boxplot, summarizing the overall differences in univariate distributions. For a high-utility synthetic dataset, the median Hellinger distance across all features is expected to be close to 0 and the variation to be small (this indicates that the synthetic data replicates the distribution of each variable in the real data accurately). Figure 4.7 shows an illustrative example of two such boxplots, where the Hellinger distance is computed as a percent.

For our purposes, it is important to make some adjustments to this approach. Since we will be dealing with binary classification datasets and our focus is on the minority class, we could



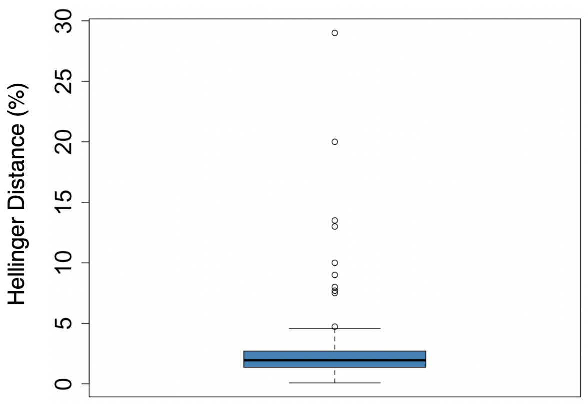


Figure 4.7: Boxplots depicting the Hellinger distance as a percent. Each data point corresponds to the Hellinger distance (as a percent) between a feature in the original data and the same feature in the synthetic data.

restrict ourselves just to the minority class and compare the distribution between the features only of the minority class samples (from the original and synthetic data). However, we believe it is important to compare the minority class-generated samples both with the majority and minority class original observations. The reason is that the univariate distributions can be very similar, which in turn can lead the synthesizer to struggle with the distinction between classes. As such, this comparison can show if the minority class generated samples are more similar to the minority class original data or to the majority class original data. Of course, it is expected that the resemblance is higher in the minority class samples, otherwise, it would indicate that the synthesizer did not learn properly and, thus, did not generate good-quality data.

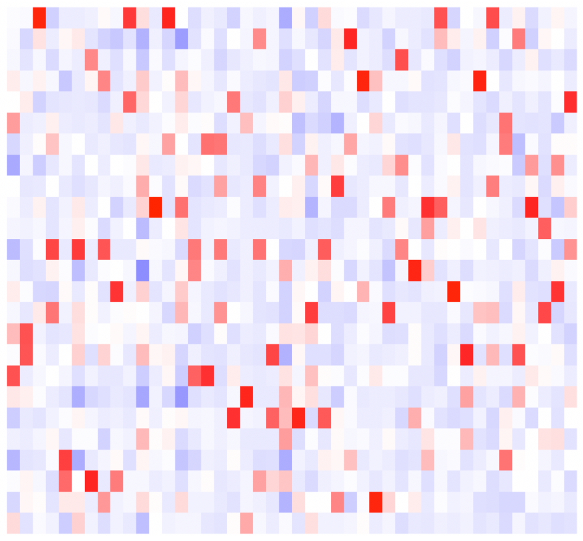
4.1.2 Bivariate Analysis

Regarding the bivariate analysis, it consists in computing the absolute difference in correlations between all feature pairs in the real and synthetic data. That is, for each pair of variables in the synthetic data, their correlation is measured; the same is performed for the original data; then, the absolute differences for the same pair of variables (the correlation of one pair was computed in the synthetic data and the other in the real data) are calculated. If the synthetic data kept the correlations between variables, then the differences will be close to zero[2].

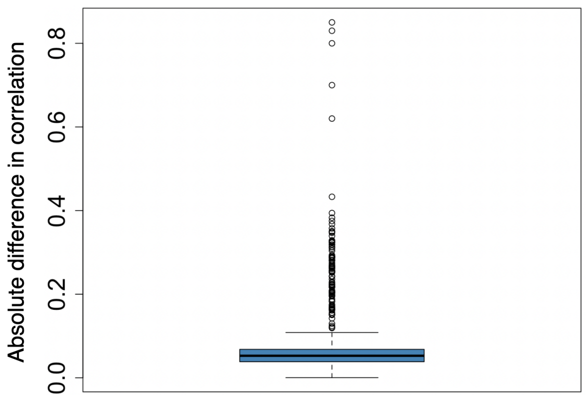
Furthermore, it is important to have in mind for which types of variables the correlations are being computed – one cannot use the same correlation metric to measure the correlation between two numerical variables and to measure the correlation between one categorical and one numerical variable. To allow for a simple interpretation of the results, all the correlation metrics used should be scaled to the $[0, 1]$ interval (or to any other interval, as long as they are all on the

[2]As in the univariate analysis, the synthetic data should not be a copy of the original data. Thus, the differences should not be all zero. Small differences indicate that the synthetic data kept the correlations between variables without overfitting the original data.

same scale). As in the univariate case, to represent the difference in correlations in a concise manner, they can be plotted on a boxplot across all possible pairwise relationships, or they can be plotted in a heatmap. Figure 4.8 shows a depiction of such a boxplot and heatmap.



(a) Heatmap.



(b) Boxplot.

Figure 4.8: Illustration of the absolute difference in correlations. The left plot shows a heatmap, whereas the one on the right is a boxplot.

Once again, in the case of having a binary (or multiple) classification dataset, this analysis can be also applied to each of the classes (as long as there are not many classes). This can be very useful to understand in which class(es) the synthesizer is having difficulties capturing the data distribution. For example, imagine that we have a binary and extremely imbalanced dataset. If we apply this approach to all the dataset, we may find the differences in correlations to be very low. Nonetheless, this may be misleading, as the minority class samples have a low contribution to these differences. By applying this approach to both classes, however, one can detect such problems.

4.1.3 Multivariate Analysis

For the multivariate analysis, it is used an *all models test*. Essentially, since it is not known *a priori* what an actual analyst would want to do with the generated dataset, determining if the real and synthetic data have similar predictive ability using multivariate models can be achieved by building classification models with every variable in the dataset as a target variable. To compute the performance of the models, the Area Under the Receiver Operating Characteristics Curve (AUROC) can be computed.

The authors propose using 10-fold cross-validation where, for each fold, the AUROC is computed. Nevertheless, we argue that a 10-fold cross-validation can be inappropriate given the dataset. For example, in the case where the dataset has few observations, 5- or 3-fold cross-validation might be more adequate. On the other extreme, if the dataset has a large number of data points, 10-fold cross-validation might not be applicable due to high training times and, thus, using fewer folds might be more suitable. Besides, if the variability of samples in the dataset is low, the 10-fold cross-validation may not bring advantages to using fewer folds. On the other hand, if the variability is high, the use of more folds might be useful.

After computing the AUROC for every fold the average is taken to compute the overall AUROC. This process is performed considering every variable as an outcome (if the dataset has n variables, the process is repeated n times, each time considering a different target variable). Moreover, since the aim is to compare the synthetic data with the real data, this has to be done both in the real and synthetic data. To guarantee that all models can be summarized in a consistent way, continuous target variables can be discretized to build classification models. Finally, the absolute difference between the AUROC values (in the real and synthetic data) is computed. Small values indicate that the models have similar performances, indicating similar predictive ability and that the models trained using the synthetic data will provide the same conclusion when applied to real data as models that were trained using real data. Once again, boxplots can be used to represent the absolute differences in AUROC values in a concise manner.

We argue that, depending on the particular use case, the AUROC might not be the best metric to consider. For example, in the case where all the variables in the dataset are numerical, it does not make much sense to discretize one at a time when they are considered the target variable. Instead, a more appropriate metric for regression can be used (such as the mean squared error or the R^2). Moreover, if one knows the

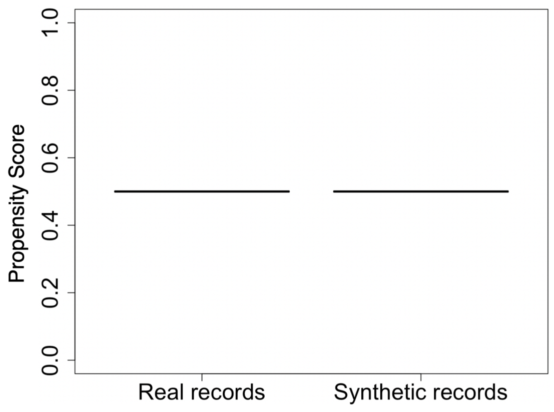
4.1.4 Distinguishability

The last utility metric addressed in this framework is distinguishability, which is another way to compare real and synthetic data in a multivariate manner. The idea is to build a model to distinguish between real and synthetic data. Hence, a binary outcome variable is assigned to each record – a 1 indicates that the observation is a real record and a 0 indicates it is a synthetic record (or vice-versa). Then, using 10-fold cross-validation[3], a classification model is trained to discern between real and synthetic samples. The classifier outputs a probability – if the probability is closer to 1, then it is predicting that a record is real, and if the probability is closer to 0, it is predicting that a record is synthetic. This is, effectively, a *propensity score* for every record. Once again, the propensity scores can be summarised in a boxplot.

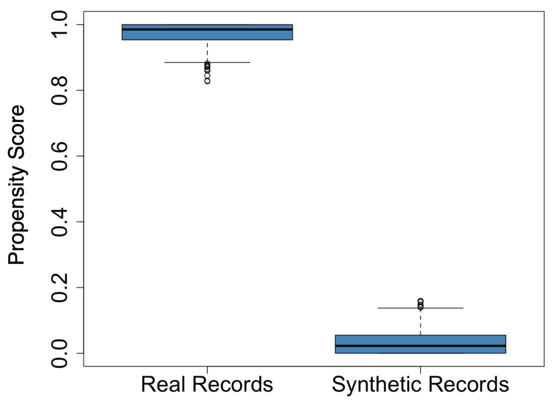
When the two datasets are exactly the same, there will be no distinguishability between them and, thus, the propensity score of every record will be 0.5 as the classifier will not be able to distinguish any observation (see figure 4.9a). On the other extreme, if the two datasets are completely different the classifier will be able to easily distinguish every record (see figure 4.9b). Of course, in a real scenario, the datasets will fall somewhere between these two extremes.

[3]in the previous section we discussed why 10-fold cross-validation might not be the best number of folds to use – the same arguments apply here.

Yet again, we argue that this approach can be applied to each of the classes (if the dataset is a classification one, of course). Moreover, if the in the presence of a classification dataset, one can, instead of providing a binary label, provide a label that indicates the class and if it is a real or a generated sample (*e.g.*, 1_{real}, $1_{synthetic}$, 2_{real}, $2_{synthetic}$, ..., n_{real}, $n_{synthetic}$, $n \geq 2$). Then, like before, the classifier will classify each sample. However, this time we have not only information regarding if the sample was predicted to be real or generated, but also the predicted class. This might be helpful to assess if some class is not being well generated.



(a) An example of distinguishability using propensity scores when there is no difference between real and synthetic data.



(b) An example of distinguishability using propensity scores when there is almost a perfect difference between real and synthetic data.

Figure 4.9: Examples of the distribution of propensity scores on two extreme cases – when the datasets are an exact copy (left plot) and when they are completely different (right plot).

The authors go a step further and propose a way of summarising the propensity scores across all records (they call them the propensity score for synthesis or PSS).

PSS1: mean square difference between the propensity score and the 0.5 value;

PSS2: converting the propensity score into a binary prediction;

PSS3: mean square difference between the propensity score and the actual 0/1 label of a record.

They show their preference for the PSS1, but note that, in practice, all three methods will provide similar conclusions. One way to interpret the PSS1 score is to split the range into quintiles since the PSS1 varies between 0 and 0.25. Ideally, the PSS1 score should be at quintile 1 (or at most at quintile 2) to ensure that the utility of the dataset is adequate. It is also easy to compare the distinguishability of different synthesis methods and datasets with the PSS. Figure 4.10 shows the PSS1 range split into five quintiles.

Overall, this utility framework is very encompassing, including univariate, bivariate, and multivariate analysis, as well as a distinguishability metric. Furthermore, it is built in such a way that tries to be very generic, providing broadly useful utility indicators when future analysis

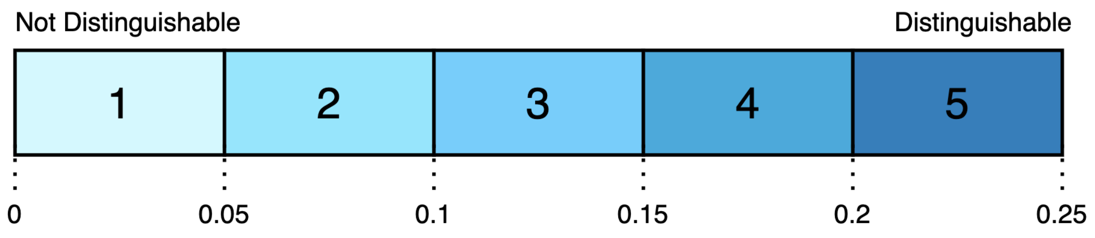


Figure 4.10: The PSS1 range can be split into quintiles.

plans are unknown. However, depending on the specific purpose of the synthetic data, this framework may not measure what is considered to be "high-quality data" for one's particular case. Furthermore, we have argued about all the 4 metrics, showing our thoughts on why something might not be adequate to a particular case or might be improved in some way.

4.2 Thoughts on the Methods

In this chapter, seven evaluation techniques were examined – descriptive statistics, graphical representations, machine learning efficacy, GAN-train, GAN-test, the (α-Precision, β-Recall, Authenticity) metric and the utility framework. It is always good not to use only one measure and to combine at least two of them. For example, as mentioned earlier, the descriptive statistics of a generated sample may be similar to those of the real data, but the distribution of data points may be very different. Or the machine learning efficacy might provide similar values for the models trained in the real data and those trained in the generated data, but their descriptive statistics, or graphical representations, may be very different.

Evaluating synthetic data is challenging and depends heavily on the problem at hand. Sometimes generating synthetic data can be useful to better train a classifier when there is a lack of data. In other cases, the problem might be to create simulated realities for a video game. The definition of "high-quality samples" is likely to be different in the two cases. In the first scenario, the synthetic data must be very similar to the original data for the classifier to learn a reasonable model of the real world. Therefore, the synthetic data must be closely scrutinized, and various evaluation techniques need to be used. In the latter case, the generated data need not be plausible in the human world, and less stringent criteria can be used to evaluate the quality of the samples.

Even for problems of a similar nature, the evaluation techniques may be different. Suppose there are two different classification tasks. The first is to classify a patient with cancer as "benign" or "malignant". The second task is to classify the sex of an unborn child as "male" or "female". In the first task, it is critical to generate extremely high-quality synthetic data to improve the classifiers. The data must be highly plausible and truly represent the real world. Failing to generate trustworthy synthetic data might lead doctors not to diagnose a patient with a malignant cancer, which can have serious consequences for the patient (and also for the doctor). Therefore, multiple evaluation techniques must be used to be sure that the generated data will help in the classification task and not jeopardize it.

In the second scenario, evaluation techniques to assess the quality of the generated samples may not need to be as rigorous. Improving the performance of the classifier might be useful even if it is with samples of intermediate quality so it is not necessary to analyze the synthetic data in detail. Whether the unborn child is classified as "female" or "male" does not have as much impact as a tumor being "benign" or "malignant".

In the next chapter, we generate synthetic data using the GAN structures described in section 3.4 and evaluate them using the utility framework proposed in this chapter.

Chapter 5

Results

Having laid out the necessary theory in the chapters 2, 3, and 4, we are now in a position to use Generative Adversarial Networks (GANs) to generate synthetic data, evaluate it, and perform data augmentation with it. The goal of this chapter is therefore to generate synthetic data. We use two well-known public datasets, the Adult dataset[1] and the LIAR-PLUS dataset[2] and use two of the GAN architectures shown in section 3.4. In addition, we evaluate the quality of our data using the utility framework described in section 4 and perform data augmentation using the generated data.

5.1 Datasets

The Adult dataset is a public dataset with information from a 1994 census database. It has 48 842 records, 14 attributes, and a target variable indicating whether the individual (each record is an individual) earns more than $\$50K$ per year or not. The Adult dataset is imbalanced, with only 11687 of the individuals earning more than $\$50K$ (roughly 24% of the observations).

The LIAR-PLUS dataset is an extension of the LIAR dataset, a publicly available dataset for fake news detection. It has 10 242 rows, 8 features, none of which is numerical, and six categories in the target variable, namely *true*, *mostly-true*, *half-true*, *barely-true*, *false* and *pants-on-fire*. As the focus of this work is to increase the performance of Machine Learning (ML) models related to Fake News detection, especially in the scenario where there is a data scarcity of the Fake News class, the following strategy was used.

(a) Only 75 random observations of the class *false* were kept.

(b) Only 75 random observations of the class *pants-fire* were kept.

(c) The 150 observations (75 + 75) were considered to be just one class – the fake news class.

[1] https://archive.ics.uci.edu/ml/datasets/adult
[2] https://github.com/Tariq60/LIAR-PLUS

(d) All the 1676 observations of the *true* class were kept.

In these conditions, we are in the presence of an imbalanced dataset, which is the focus of our study.

5.2 Synthetic Data Generation

For the synthetic data generation, we have used the CTGAN and the TabFairGAN. The TGAN, although mentioned in section 3.4, is not as powerful as the CTGAN nor the TabFairGAN (in fact, the CTGAN is an improvement over the TGAN), so it was not used. Moreover, from the author's own experience, the code implementation of the CTGAN is much more robust than the one of the TGAN and allows for the replication of results.

Table 5.1: Hyperparameters used for the CTGAN architectures.

Noise dimension	64, 128
Size of the output samples for each one of the Residuals	(512, 512), (256, 256)
Size of the output samples for each one of the Discriminator Layers	(512, 512), (256, 256)
Generator learning rate	0.0002
Discriminator learning rate	0.0002
Number of discriminator updates to do for each generator update	1
Whether to use log frequency of categorical variables	True
Number of samples to group together when applying the discriminator	16

Due to the way these models were implemented by their authors, only the CTGAN[3] allowed us to change its architecture (hyperparameters). As such, in the case of CTGAN, we have used 8 different architectures (see table 5.1), whereas in the case of the TabFairGAN[4] we were limited to the default values. Throughout this chapter we make reference to the CTGAN architectures, so we have named them as follows for easy reference in table 5.2. We note that the generator dimension and the discriminator dimension are the size of the output samples for each one of the residuals and the size of the output samples for each one of the discriminator layers, respectively.

5.3 Synthetic Data Evaluation

The goal of this section is to analyze the data quality of the synthetic data generated using the Adult dataset. For such purposes, we have used the data utility framework explained in chapter 4, as it is generic and encompassing. Given this is a rather extensive analysis, we have chosen to only perform it in the Adult dataset. The main reason is that the LIAR-PLUS dataset has several text variables, causing this framework to not be as easily applicable. Nevertheless,

[3]The code of the CTGAN can be seen at https://sdv.dev/SDV/api_reference/tabular/ctgan.html

[4]The code of the TabFairGAN can be seen at https://github.com/amirarsalan90/TabFairGAN

Table 5.2: Summary table of the eight CTGAN architectures.

Architecture	Noise Dimension	Generator Dimension	Discriminator Dimension
CTGAN 0	64	(512, 512)	(512, 512)
CTGAN 1	64	(512, 512)	(256, 256)
CTGAN 2	64	(256, 256)	(512, 512)
CTGAN 3	64	(256, 256)	(256, 256)
CTGAN 4	128	(512, 512)	(512, 512)
CTGAN 5	128	(512, 512)	(256, 256)
CTGAN 6	128	(256, 256)	(512, 512)
CTGAN 7	128	(256, 256)	(256, 256)

the following paragraphs show the potential of this evaluation framework, which is more crucial than the dataset that is being used.

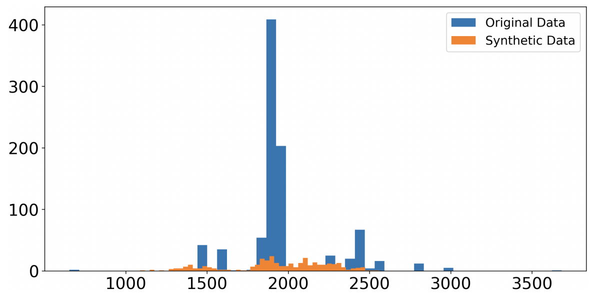
Before carrying on with the evaluation it is important to note that:

(a) the features *capital-gain* and *capital-loss* were not included in this analysis because they were poorly generated by the GAN models. The real distribution is completely different from the generated distribution (see Figure 5.1). Therefore, they were not considered because they would interfere with the analysis given their poor results.

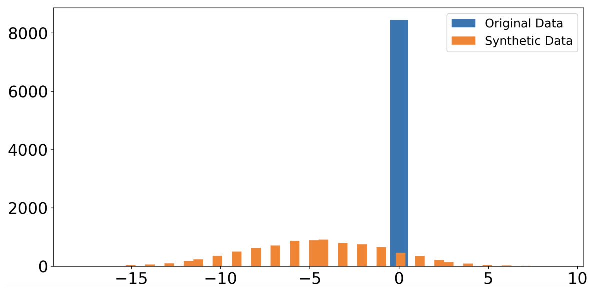
(b) given the fact that we are analyzing synthetic data of trained GANs, we are (quite reasonably) assuming that the statistical properties of a sample do not depend on its size – samples with sizes 1000 and 10000 have the same statistical properties. This assumption seems plausible, as the model has learned a certain distribution from which it samples the synthetic data. Thus, the sample size does not influence the statistical properties of the generated data.

(c) since we have only generated samples of the minority class, we have only compared the generated samples with the original minority class samples, as that is the distribution we are trying to capture.

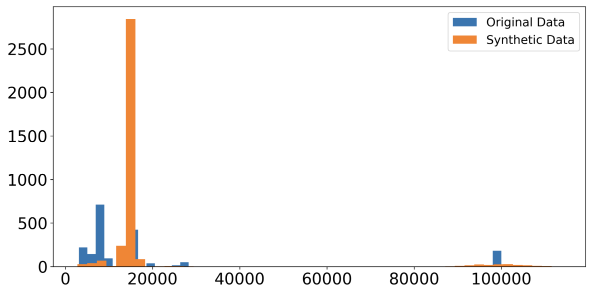
Having this in mind, we proceeded to the synthetic data evaluation using the utility framework already described in chapter 4.



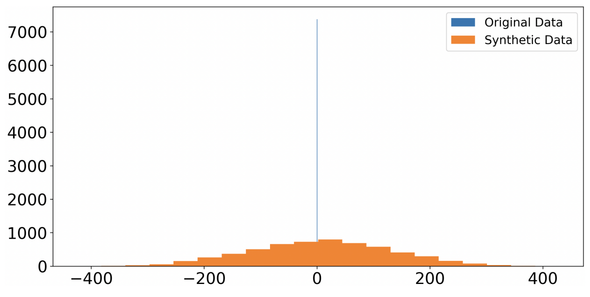
(a) Histogram of the variable *capital-loss* in the synthetic and real data for values ≥ 500.



(b) Histogram of the variable *capital-loss* in the synthetic and real data for values < 500.



(c) Histogram of the variable *capital-gain* in the synthetic and real data for values ≥ 500.



(d) Histogram of the variable *capital-gain* in the synthetic and real data for values < 500.

Figure 5.1: Histogram for the variables *capital-loss* (upper plots) and *capital-gain* (lower plots). For each variable, there are two histograms, one for values equal to or greater than 500 and one for values less than 500 (for visualization purposes). The blue bars correspond to the original data, whereas the orange ones correspond to the synthetic data. The synthetic data in the plots was drawn from the CTGAN with the architecture 0, but the results were similar in all the other architectures.

5.3.1 Univariate Analysis

For the univariate analysis, we have used the Hellinger distance to compare the distributions between the real and the generated features. Figure 5.2 shows the several resulting boxplots. In the boxplots below each data point is the Hellinger distance between the observations of a variable in the original data and the observations of that same variable in the synthetic dataset. Each variable in the synthetic dataset was compared with the original minority class data (same class) and the majority class (opposite class). Of course, we expect the synthetic univariate distributions to be more similar to the same class original data distribution, otherwise the synthetic data generation would have been terrible.

From the figure 5.2, it can be seen that the univariate distributions in all plots are more similar to the original minority class data than to the majority class data. It can also be seen that CTGAN 2 has a low median and all points are below 0.4, indicating that the univariate distributions were well preserved in the generated data. CTGAN 3 also produced good results, but has a visible outlier. The same is true for architectures 4 and 7 and TabFairGAN. Overall, CTGAN 2 seems to retain the univariate distributions best.

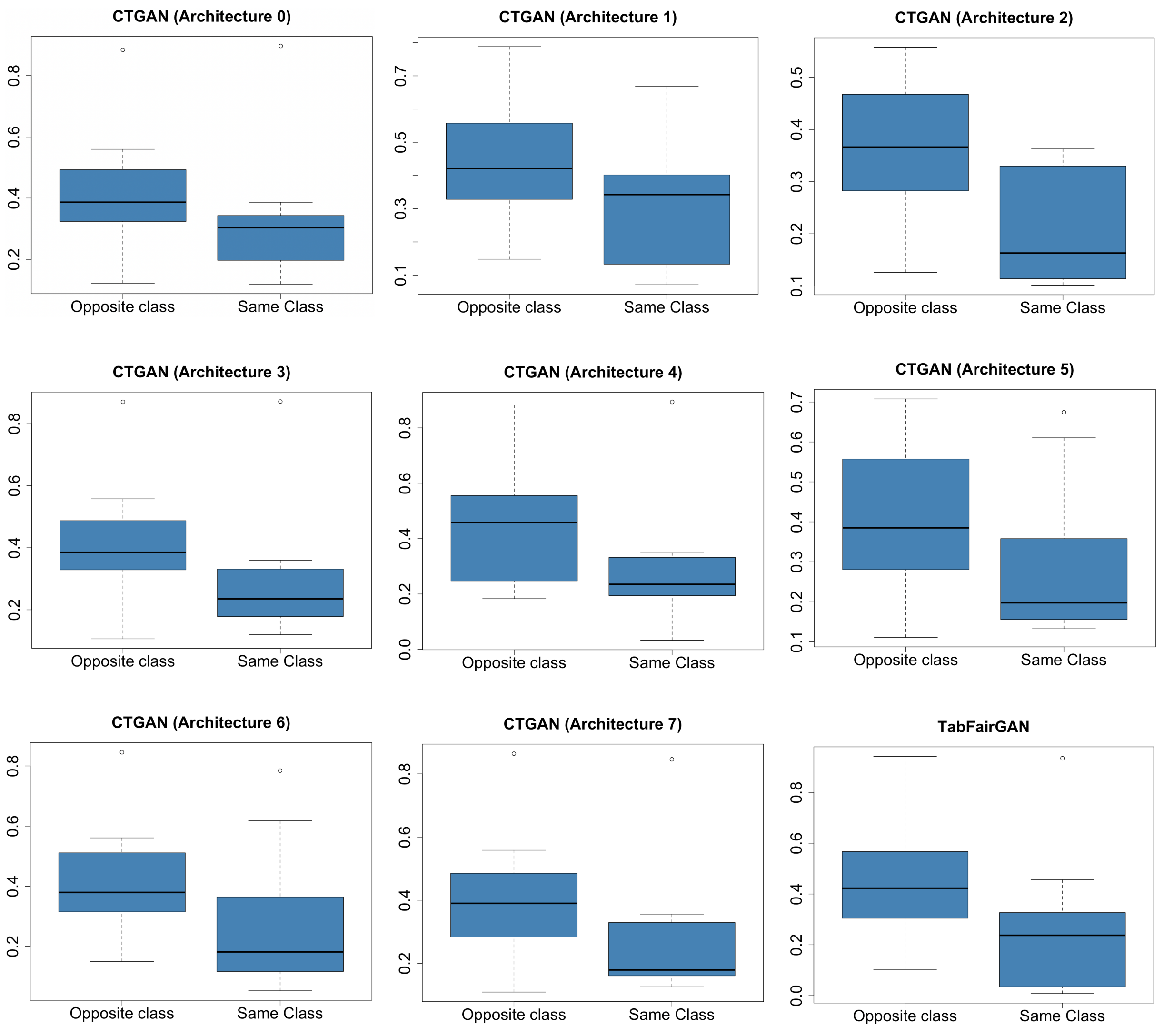


Figure 5.2: Boxplots for each GAN architecture comparing the univariate distributions of the generated samples both with the minority as well as the majority class original samples.

5.3.2 Bivariate Analysis

Regarding the bivariate analysis, we have used the *Pearson correlation* for correlations between continuous variables, *Cramer's V* for correlations between categorical variables and the *correlation ratio* for correlations between continuous and categorical variables. As it would be more difficult to compare each heatmap of the difference in correlations, once again we use boxplots (see figure 5.3).

From the analysis of the boxplots, the TabFairGAN seems to provide the best results, with a small range of values and a very low median. Indeed, it is quite better than the other architectures. This indicates that it did a good job maintaining the correlations between every variable pair.

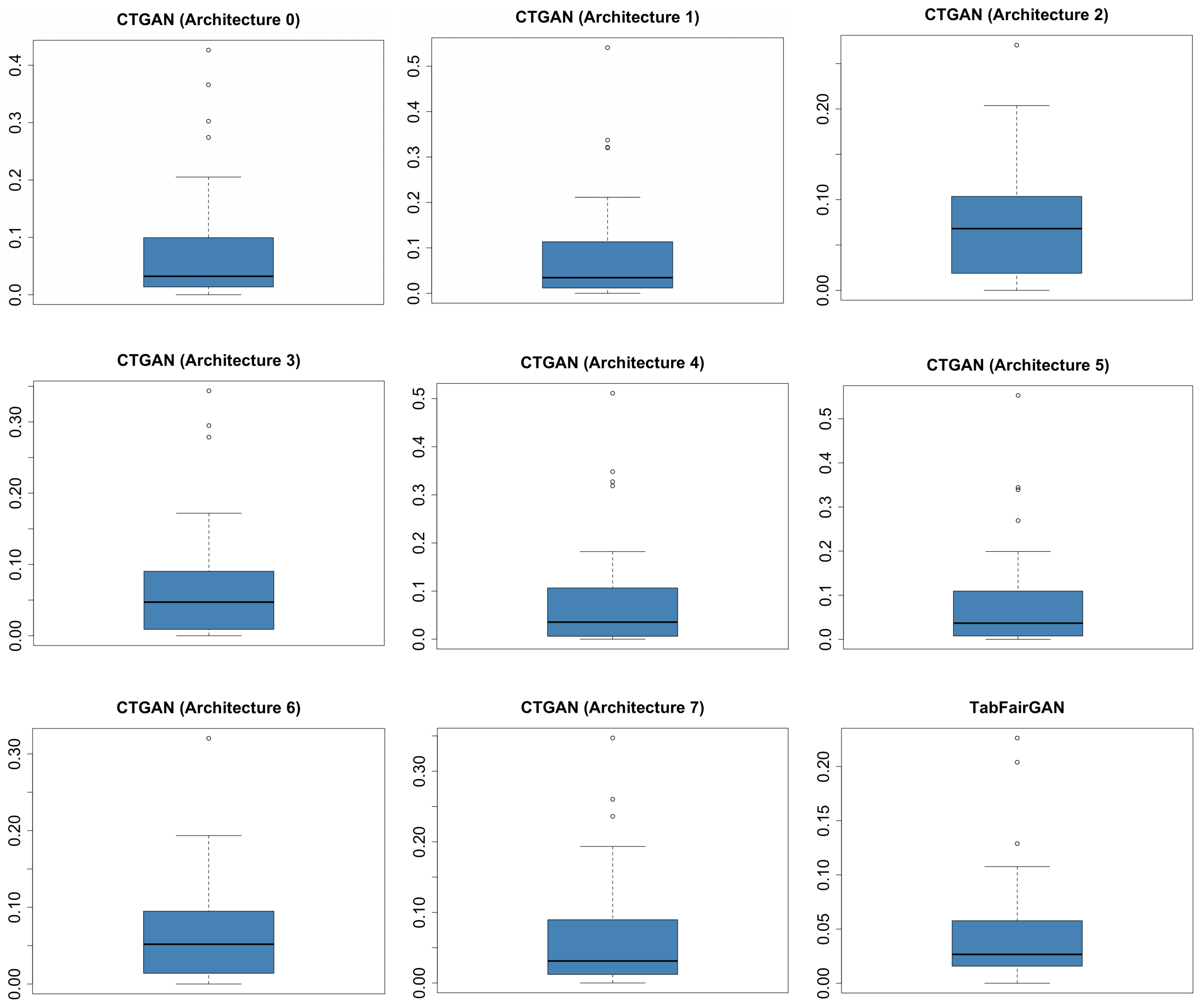


Figure 5.3: Boxplots for each GAN architecture showing the difference in pair-wise correlations between the generated samples and the original samples.

5.3.3 Multivariate Analysis

For each synthetic dataset, we have considered each variable as a target variable, used 5-fold cross-validation to train a Decision Tree, and computed the mean Area Under the Receiver Operating Characteristics Curve (AUROC) across the 5 folds. Moreover, the variables *age* and *hours-week* were discretized, so that we could use a classifier and, thus, compute the AUROC for every outcome variable. Figure 5.4 shows several boxplots with the absolute differences of the mean AUROC across all folds between the synthetic data and the original data, for each synthetic dataset and outcome variable.

In this case, it is obvious that the TabFairGAN has provided considerably better results than all the CTGAN architectures. With the smallest range and inter-quartile range, as well as the lowest median, the TabFairGAN as provided outstanding results in the multivariate analysis.

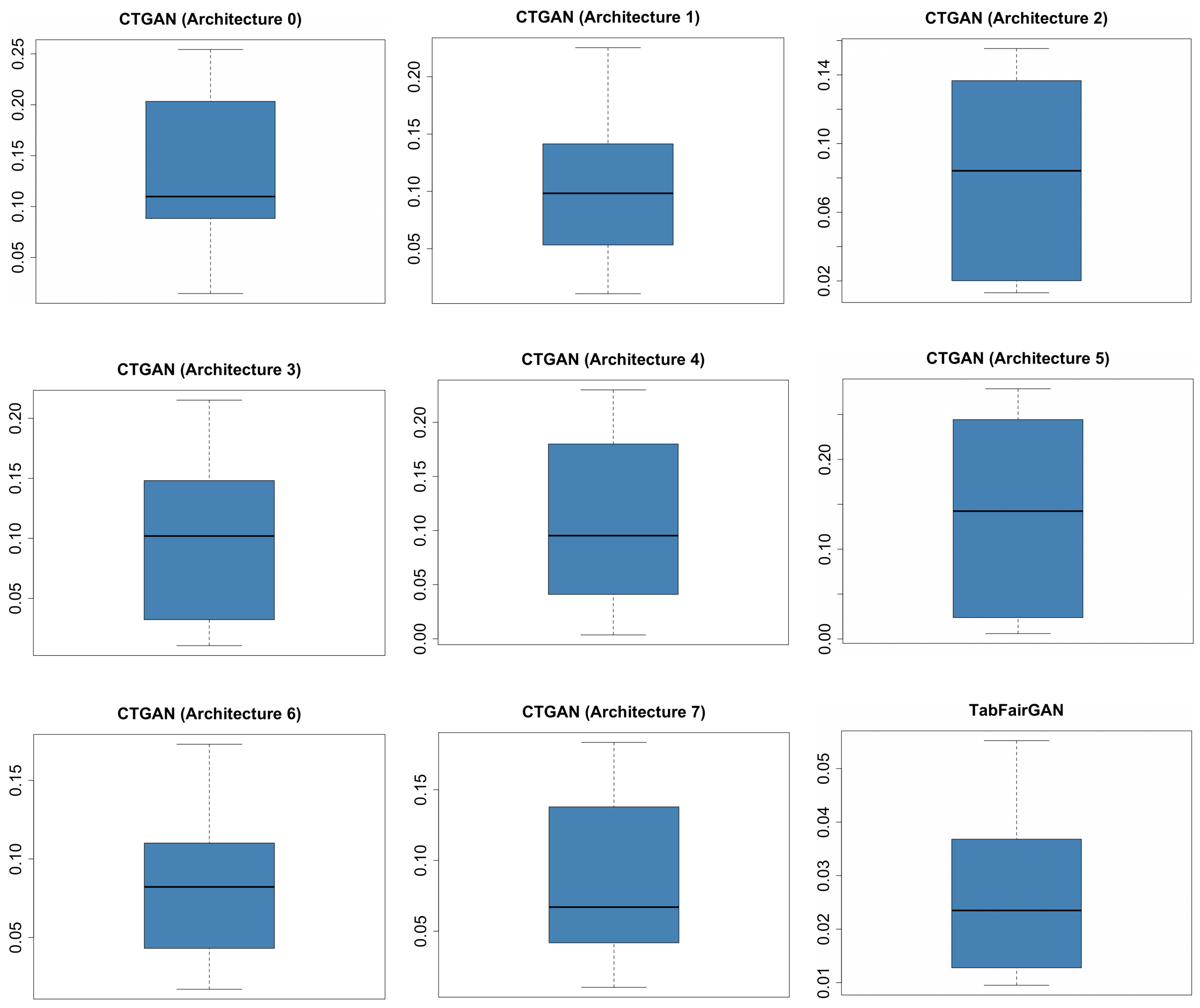


Figure 5.4: Boxplots for each GAN architecture comparing the differences in the mean AUROC between the generated samples and the original samples.

5.3.4 Distinguishability

Regarding the distinguishability, each record was assigned a binary label (0 if it was a synthetic record and 1 otherwise) and an XGBoost model was trained using 10-fold cross-validation. Once again, boxplots were used to assess the data quality (see figure 5.5).

To further aid our analysis, we have used the PSS1 score to summarise the propensity scores across all records. Table 5.3 shows the PSS1 (already explained in subsection 4.1.4), as well as the corresponding quintile (also shown in subsection 4.1.4) for every synthetic dataset.

From table 5.3 it becomes clear that the TabFairGAN, once again, has provided better results than all the other architectures. This indicates that the utility of the TabFairGAN generated data is quite high.

Overall, the TabFairGAN provided the best results. It was the best in terms of distin-

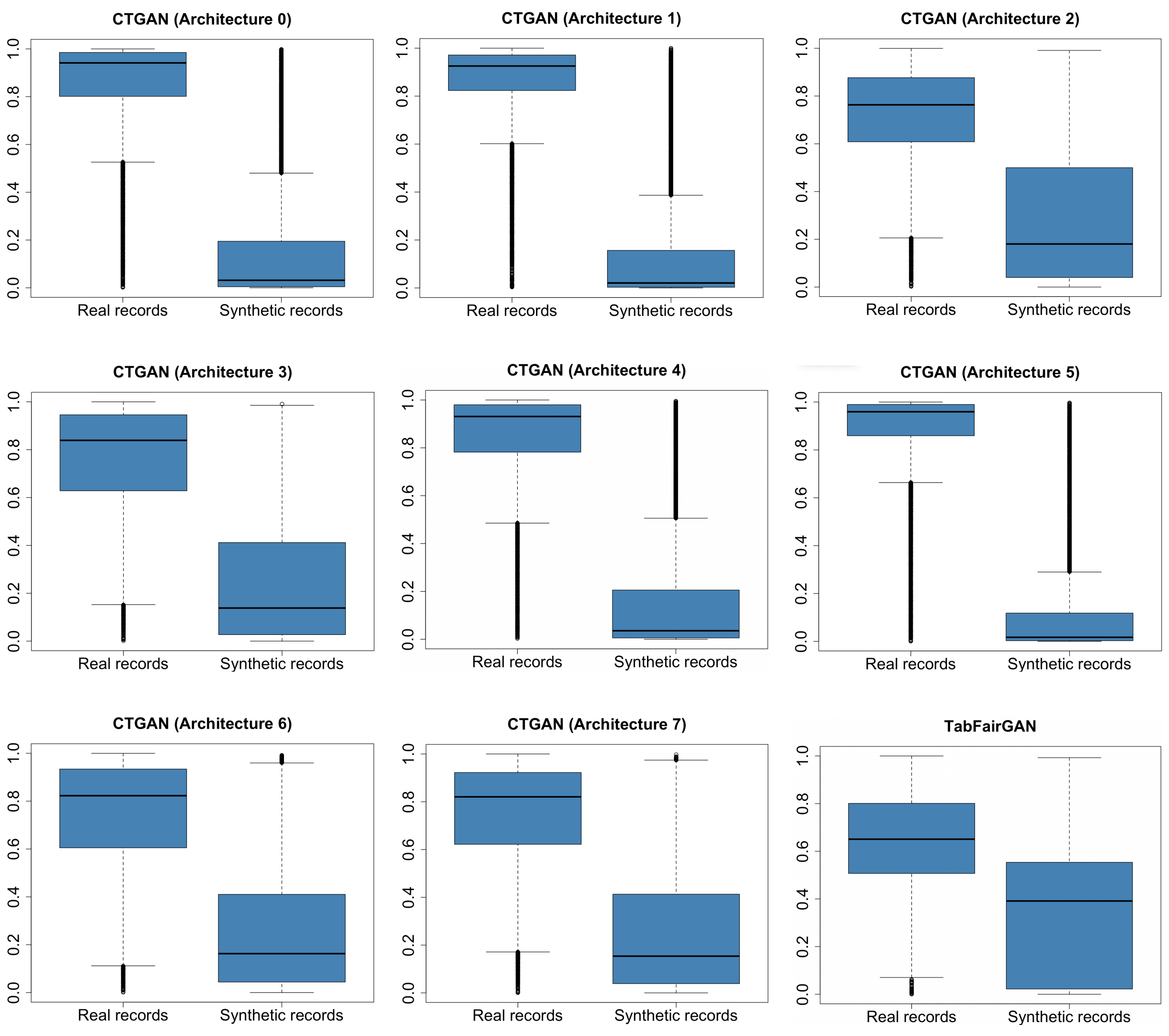


Figure 5.5: Boxplots for each GAN architecture comparing the propensity scores of the generated samples and the original samples.

Table 5.3: Summary table presenting the PSS1 score as well as the corresponding quintile for every synthetic dataset.

GAN	CTGAN 0	CTGAN 1	CTGAN 2	CTGAN 3	CTGAN 4	CTGAN 5	CTGAN 6	CTGAN 7	TabFairGAN
PSS1	0.1707	0.173	0.1061	0.1265	0.1676	0.1854	0.1196	0.1203	**0.0773**
Quintile	4	4	3	3	4	4	3	3	**2**

guishability and in the all-models test (multivariate analysis). In the bivariate analysis, it also provided very good values, being the best architecture of them all (it had the lowest median and the lowest interquartile range). In this bivariate analysis, other architectures provided some competition, namely CTGAN 7, for example, but the outliers in the boxplot were farther away than the ones of the TabFairGAN. Regarding the univariate analysis, the CTGAN 2 was the one that better kept the univariate distributions of variables. To better assess the TabFairGAN supremacy, a radar chart was built (see figure 5.6). Each axis represents one of the metrics used and the lines represent the GAN architectures. The axes *Univariate*, *Bivariate* and *Multivariate* correspond to the Hellinger distance, the difference in pair-wise correlations and the difference in the mean AUROC between real and generated samples, repectively. Moreover, these metrics are represented by their medians, as it would be impossible to include all the points in the radar chart.

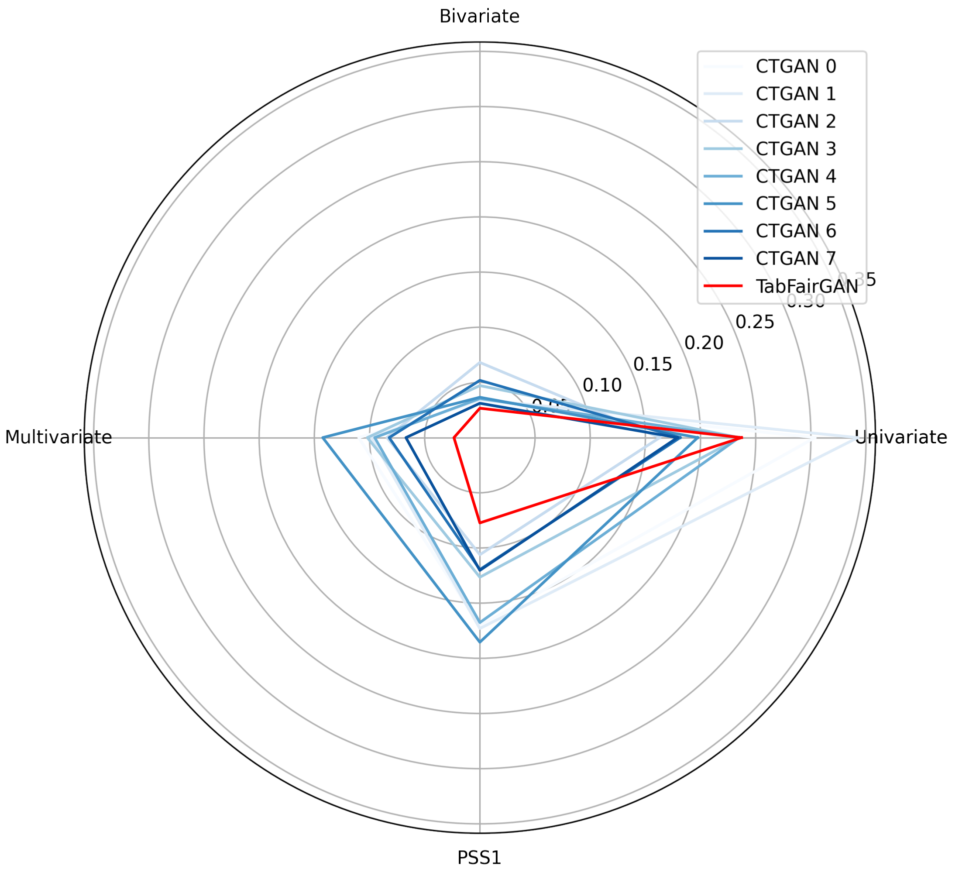


Figure 5.6: Radar chart comparing the several GAN architectures (represented by the colored lines) in terms of the 4 evaluated metrics (represented by the axis). The TabFairGAN architecture has, clearly, provided very good results as its line (the red one) is the more inward one.

In the next section, this data (as well as the LIAR-PLUS dataset) will be used for data augmentation, so that we can assess if the aid of synthetic data can aid in the performance of ML models in a classification task, in particular in a news dataset.

5.4 Data Augmentation

Now that we have described the datasets, explained what architectures are used for data generation, and used the data utility framework to evaluate the synthetic data, the next step is to perform data augmentation. Despite, the fact that the utility framework was only applied to the Adult dataset (the reasons were already mentioned), we have seen that the architectures used provided, typically, good results (the TabFairGAN having provided the best results). Thus, we can generalize to a certain extent, and assume that the quality of the generated samples of the LIAR-PLUS dataset are reasonably good.

5.4.1 Adult Dataset

The adult dataset was first split into train (80% of the observations) and test (the remaining 20%) sets. The several GAN architectures presented in section 5.2 were trained using only the minority class samples of the train set (as this is the class that is lacking observations) for 300 epochs with a batch size of 256. Once the GANs were trained, we generated a number of samples such that the minority class had 40%, 50%, . . . , 100% of the size (number of observations) of the majority class (in the original train set, the minority class was of about 31% the size of the majority class).

Next, we have trained multiple ML models – Logistic Regression, Decision Tree, Random Forest and XGBoost – using the original train set (to serve as a baseline), but also the original train set plus the synthetic minority samples generated by the trained GAN architectures (the augmented dataset). Each model was trained 3 times in each dataset (original and augmented datasets) and the mean of each 3 runs was computed to decrease the randomness involved. The models were then evaluated in the test set, which has been placed aside and is only used in this step. The accuracy, precision, recall, and F1 were measured for both classes.

To better interpret the results, we have used various plots. As all the plots were created using the same idea in mind, we shall explain them now, as this explanation will hold for all the following plots and we can avoid being repetitive (the reader can look at figure 5.7 to follow the explanation). Each figure has several subplots, where subplots in the same column refer to the same metric and subplots in the same row refer to the same ML model. The x-axis represents the percentage of minority class samples when compared to the total of majority class samples. The blue lines (dark and light blue) indicate the performance of the models when trained in the augmented datasets. The red lines (dark and light red) indicate the performance of the models when trained in the original dataset – that is why they are constant (we have just drawn the constant line for visualization purposes). We also note that not every plot was included, but only the most important ones, as it would be exhaustive to list them all.

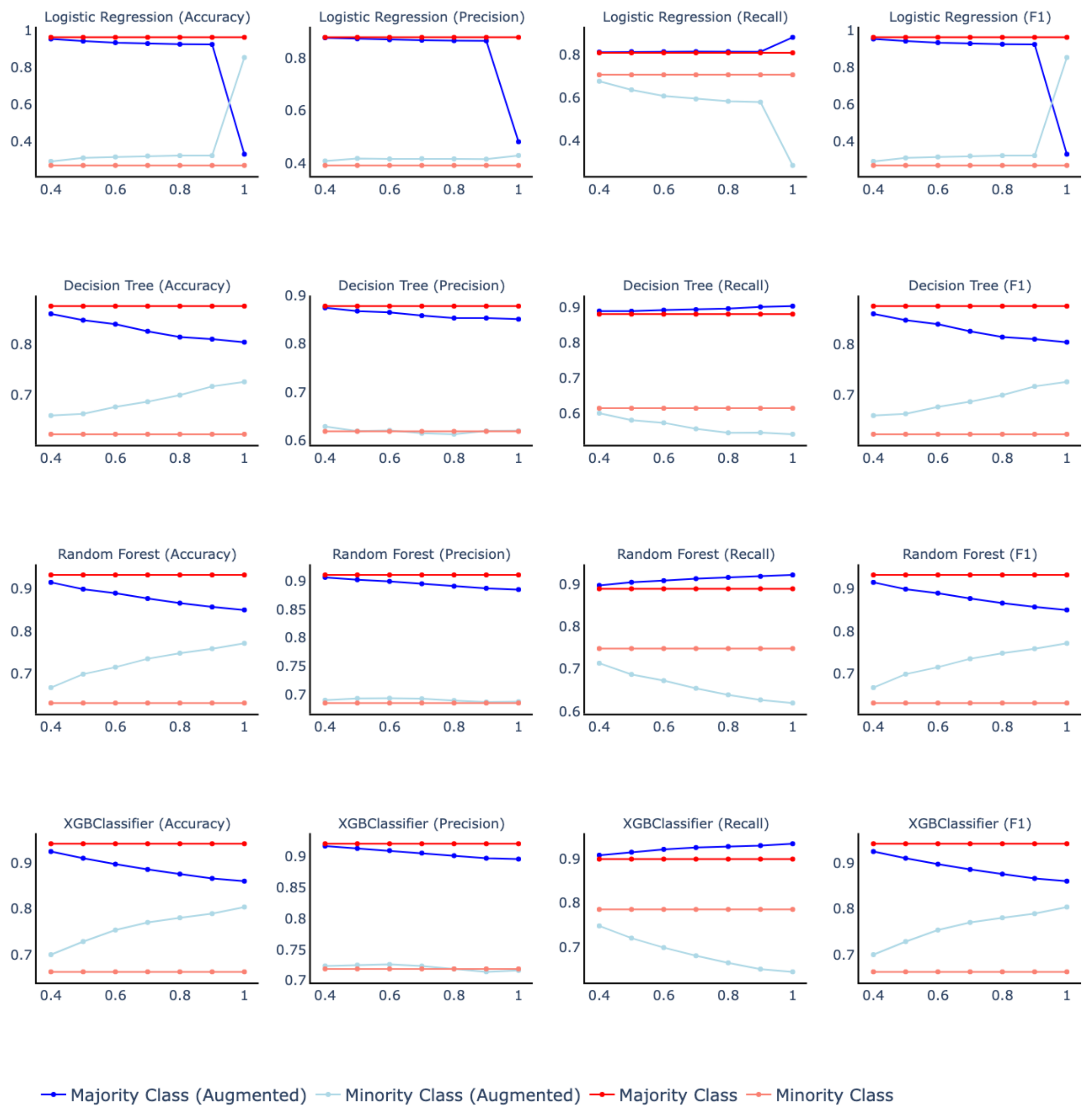


Figure 5.7: Performance comparison of the 4 models across the 4 metrics when tested in the test set and trained in the augmented dataset (TabFairGAN) as well as in the original training set.

Figure 5.7 shows the results of the data augmentation for the TabFairGAN-generated datasets. It is quite interesting to note that, even though the TabFairGAN provided the best results in the evaluation framework of the previous section, it does not provide very good results in the data augmentation task as a whole. That is, it can substantially increase the performance in the minority class, which is our focus, but also diminishes the performance in the majority class. That being said, depending on the particular use case, one might be able to compromise the performance in the majority class to obtain better results in the minority class, given the high increase of performance in the underrepresented class. This helps us answer **RQ3**, as it can be see that, indeed, good synthetic data quality can bring performance increases in the classification task for the minority class. However, the performance in the majority class may decrease, given that the synthetiser only trained with minority class samples, the focus being on the underrepresented class.

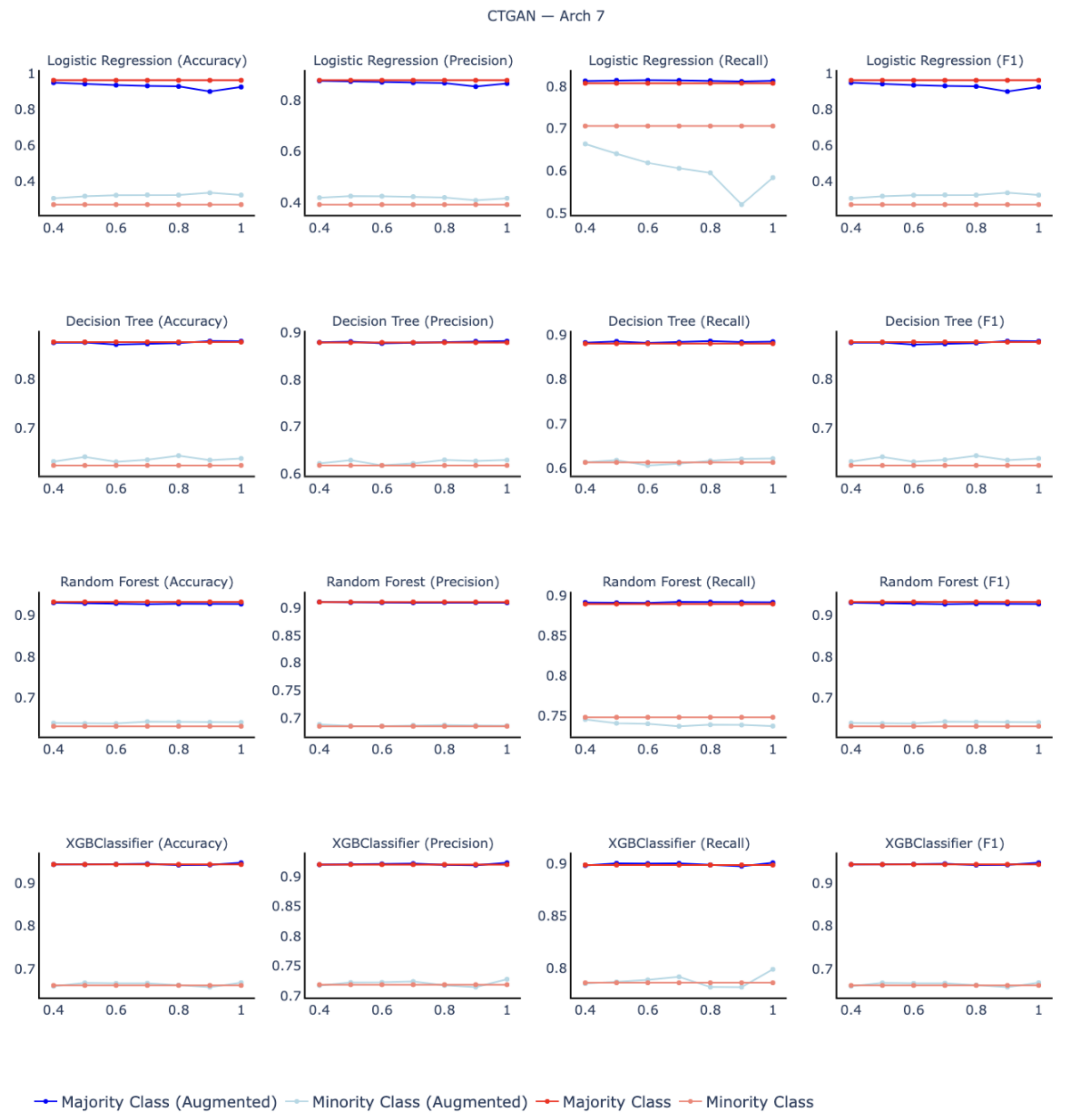


Figure 5.8: Performance comparison of the 4 models across the 4 metrics when tested in the test set and trained in the augmented dataset (CTGAN 7) as well as in the original training set (Adult dataset).

As for the CTGAN architectures, some of them provided very good results, the best of which was the CTGAN 7 – figure 5.8 shows the results for this architecture. As can be seen from the subplots, in all of them the performance in the minority class increases in the augmented dataset at almost no expense from the performance in the majority class (the exception being the recall on the Logistic Regression and in the Random Forest). In fact, in some cases, the performance in the majority class increases too.

Overall, the results were somewhat surprising. Even though the TabFairGAN provided the best results in the evaluation framework (see figure 5.6), indicating that the data was of high utility, and the generated data substantially increased the performance in the minority class, the results of the classification as a whole (that is, considering both classes) did not improve. However, the CTGAN 7, which did not provide results as good as the TabFairGAN in the evaluation framework, has done a very good job in the data augmentation task, bringing information that helped the ML models increase the minority class performance, without compromising the majority class. In fact, this type of experiment can also be seen as an evaluation technique of the quality of the generated data. It does not compare the properties of the synthetic data with those of the original data but shows to what extent can such data help ML models increase their performances.

5.4.2 LIAR-PLUS Dataset

The LIAR-PLUS dataset needed more processing, as it contained several text features. The variables *job*, *state*, *context* and *justification* had some missing values, which were replaced by the value *other*. The variable *subject* was a list of subjects, so we expanded that list into several columns (20 in total) – each column had only one subject ($subject_i$). Since there were 20 subjects and most of them had null values, we kept just the first 7 columns, as there was almost no loss of data. Furthermore, given the fact that the variables *speaker*, *job*, *state*, *party* and $subject_i$ ($\forall i \in \{0, ..., 6\}$) had several categories, it would be infeasible to one hot encode them, as the number of columns would grow very large – in the order of magnitude of the thousands. Hence, the variables were target encoded.

Finally, the variables *statement*, *context* and *justification* were actual sentences and were the variables that, most likely, contained the more important information to predict the target variable. Thus, they were processed using the Doc2Vec algorithm, which takes the sentences and maps them into a numerical space. For the *statement* and *justification* variables, a mapping into a 16-dimensional space was performed. For the *context* variable, a mapping to an 8-dimensional space was performed. The *context* variable has fewer words than the *statement* and *justification* variables, hence the lower-dimensional space: not as much information to be mapped, thus the use of fewer dimensions.

Since the LIAR-PLUS dataset was already divided into train and test sets, we kept that division. As explained above, the train was modified to only include the target classes *pants-fire*, *false* and *true*, with an intentional imbalanced target. As for the test data, we simply kept those same classes, again merging the *pants-fire* and *fake* classes into one class. The train had 1826 observations (1676 of the *true* class and 150 of the other class). The test had 1267 observations (208 of the *true* class and 1059 of the other class). We note that the test set is lacking data in the *true* class when compared to the $fake$ class. These were the proportions already in the test data, and they were not altered.

To assess the results we have used the same type of plots used in the previous section to evaluate the data augmentation in the Adult dataset, so we refrain from explaining them again.

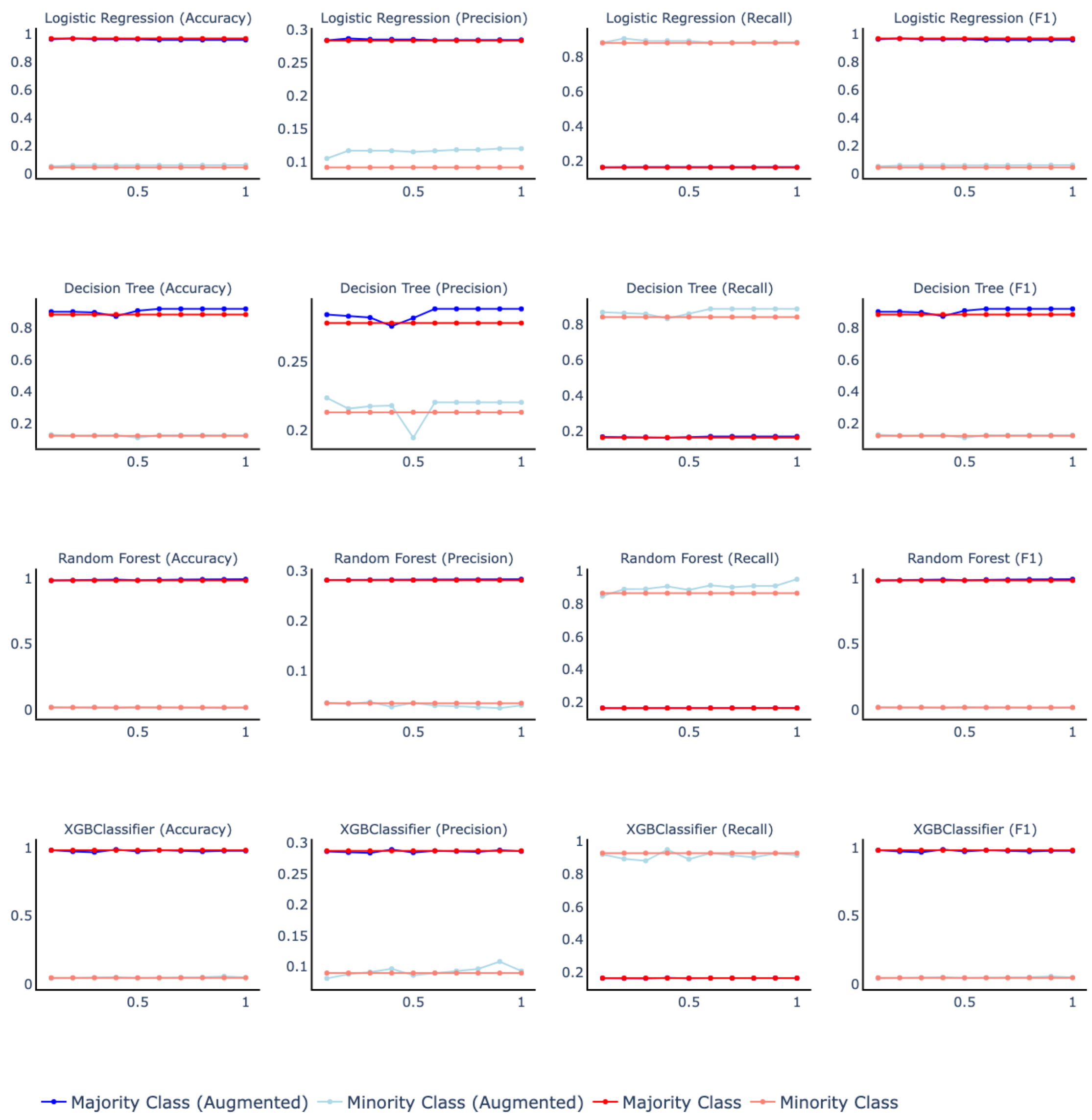


Figure 5.9: Performance comparison of the 4 models across the 4 metrics when tested in the test set and trained in the augmented dataset (TabFairGAN) as well as in the original training set (LIAR-PLUS dataset).

Figure 5.9 shows the results for the TabFairGAN which, this time, provided better results, especially regarding the Decision Tree model, where the precision and the recall metrics have increase in the minority, as well as the majority class samples. Nonetheless, the CTGAN architectures have, once again, been able to perform a better job at the data augmentation task (see figures 5.10 and 5.11).

From figure 5.10 we can observe that the CTGAN 2 was able to increase the precision and the recall across all models in the minority class, especially in the Decision Tree, where the uplift

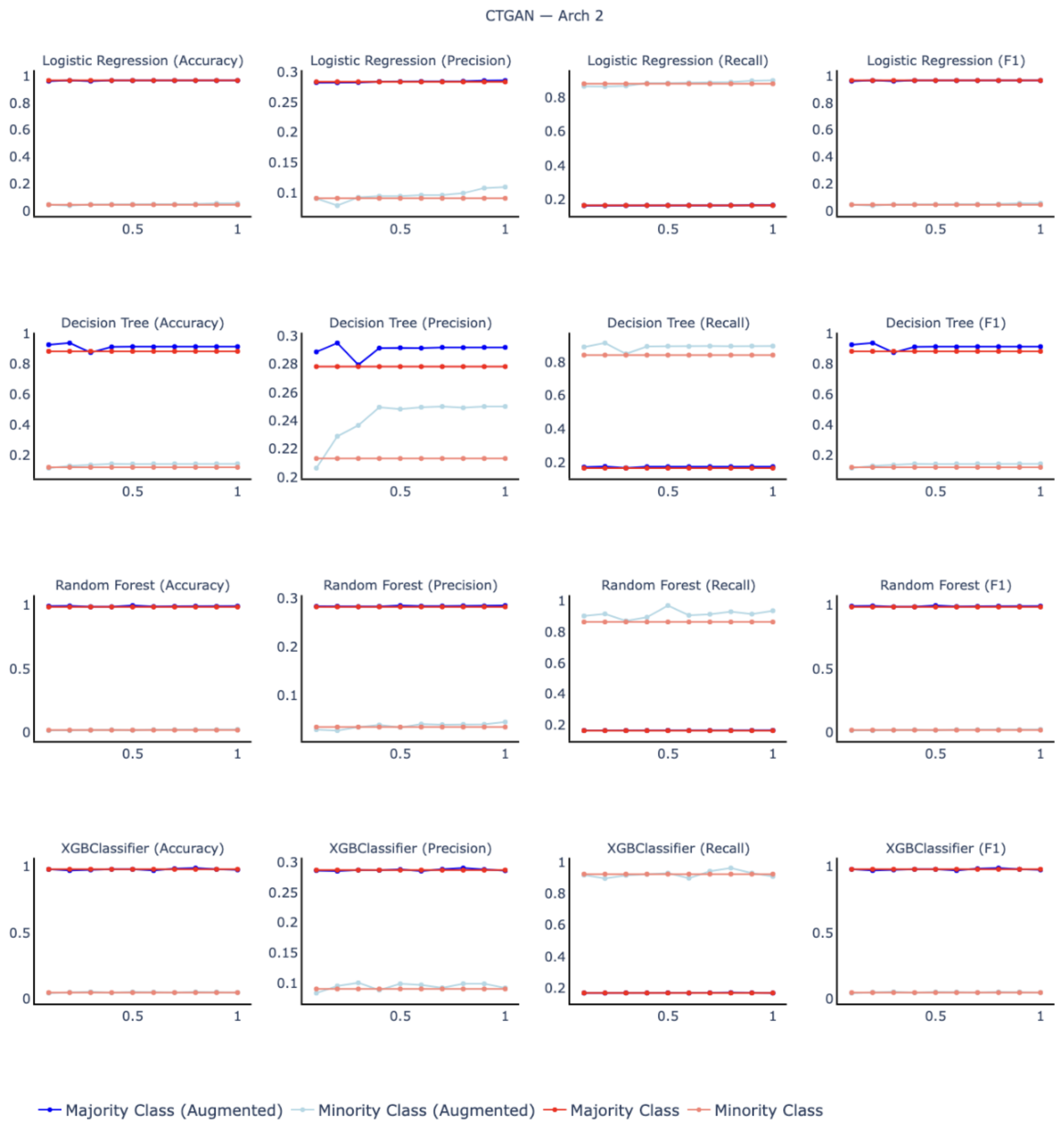


Figure 5.10: Performance comparison of the 4 models across the 4 metrics when tested in the test set and trained in the augmented dataset (CTGAN 2) as well as in the original training set (LIAR-PLUS dataset).

in the precision metric was quite high. Moreover, this increase in performance came without a cost regarding the majority class metrics.

As with the CTGAN2, the CTGAN 4 (see figure 5.11) has also provided very good results in the precision and recall metrics regarding the minority class samples, especially in the Decision Tree. Furthermore, this architecture was able to increase the precision metric in the XGBoost model, an already very powerful model. This increase in performance in the minority class came, once again, without a cost in the majority class.

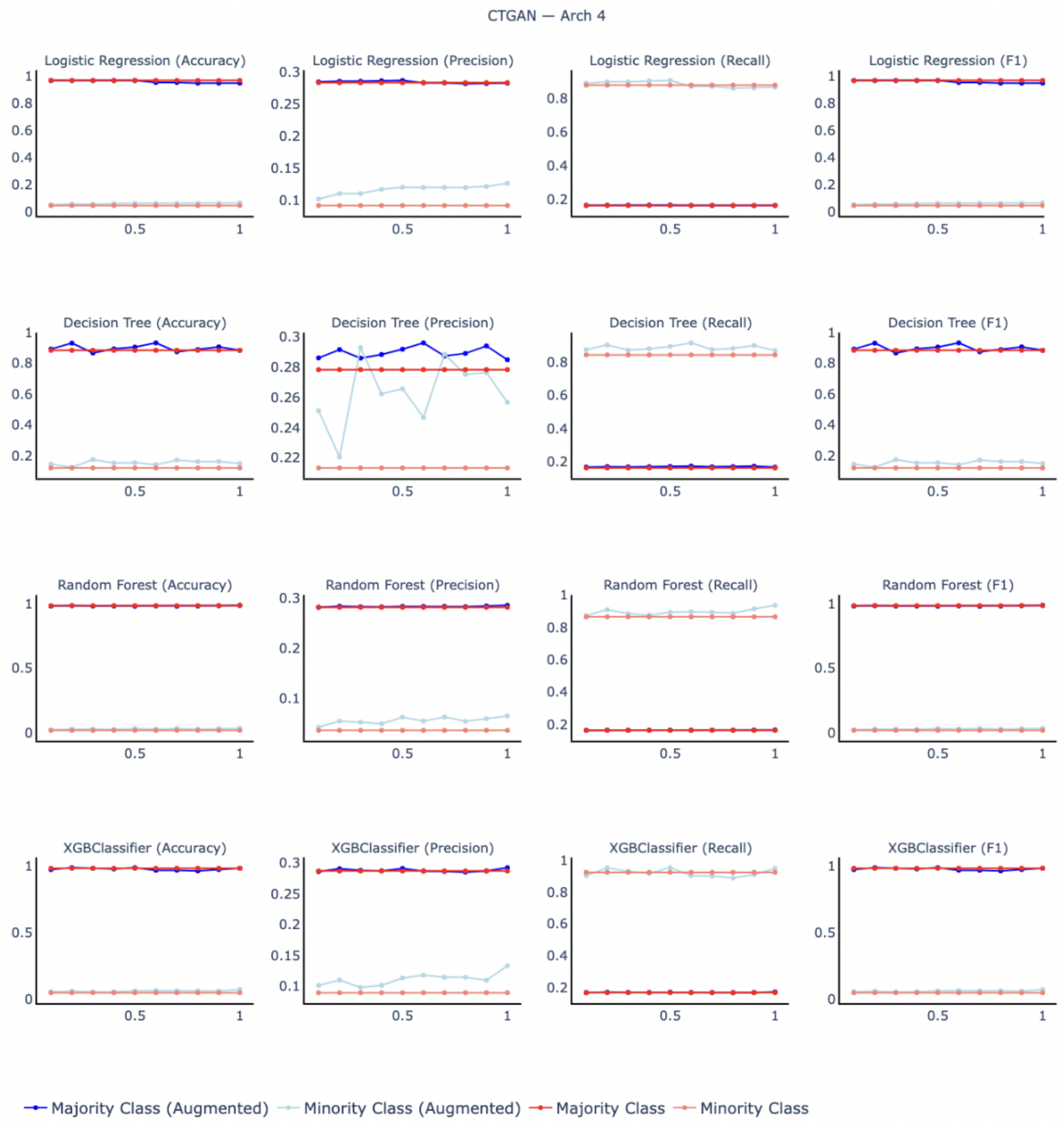


Figure 5.11: Performance comparison of the 4 models across the 4 metrics when tested in the test set and trained in the augmented dataset (CTGAN 4) as well as in the original training set (LIAR-PLUS dataset).

These results are pretty promising. On the one hand, we have increased the performance in the minority class (the Fake News class). On the other hand, this increase did not incur in decreases in the majority class. Moreover, these results help us answer **RQ1**. Given we have increased the performance in the class of interest without jeopardizing performance in the other, the results obtained support the fact that, indeed, the addition of synthetic data can aid in the identification of Fake News.

In light of the results explored in this chapter, in the next chapter, we discuss some conclusions taken from this work, some challenges faced and limitations, and possibilities for future work.

Chapter 6

Conclusion

Fake News are present in our daily lives. They change our perception of the world, influence election results and political decisions, negatively impact public health and debates, and divide people's opinions. This is by no means a recent problem, but with the rise of social media platforms, the prevalence and reach of Fake News is steadily increasing. This is a serious problem, and any tool that can help identify Fake News can be a valuable asset.

This chapter contains a general summary of the work carried out, a recapitulation of the research questions and some conclusions that have emerged. Finally, some limitations and challenges are discussed and possibilities for future work are identified.

6.1 General Summary

The work carried out in this investigation included several phases. The first was to understand what methods already exist for generating synthetic data, what their strengths and drawbacks are, and which are the most promising. In recent years, Generative Adversarial Networks (GANs) have received a lot of deserved attention. They are a strong idea and have provided very good results, especially in image generation. Nowadays, the images generated by GANs are so good that we cannot distinguish between a real image and a synthetic one.

Having understood that GANs are a powerful tool, we explored it further. First, we understood the paradigm of GAN training using an adversarial approach. With this knowledge, we examined the evolution of GANs in chronological order to better understand its improvements over the years as well as its current state. In the next step, we focused only on tabular GANs as they are the most useful for our work. In this area, they are very underrepresented compared to the image domain, making it, clearly, an understudied topic.

The next step was to understand how to evaluate the quality of the synthetic data. Until now, we have been funneling things down, from synthetic data generation methods, to GANs and to tabular GANs. But once this was settled, we needed to comprehend how to evaluate the quality of the generated samples. We researched the most useful synthetic data evaluation methods and came across an interesting idea of a data utility framework.

Now that we had all the necessary ingredients, we were able to use GANs to generate synthetic data, evaluate it, and use it for data augmentation, which we did. In these experiments, we showed that the quality of the synthetic data does not necessarily mean that using the augmented dataset will give better classification results. In addition, we have shown which GANs are more suitable for generating synthetic data and, most importantly, the potential of this approach to increase the performance of Machine Learnings (MLs) classifiers in detecting Fake News. The experiments allow us to answer the research questions raised in Chapter 1. These questions are reproduced below along with the answers obtained in this study.

1. **RQ1: Does the addition of synthetic data in the training process of ML models enable better detection of fake news?** In section 5.4, we used a public news dataset to answer this question. The experiments conducted showed that the addition of synthetic data can have a positive impact on the performance of the ML models in detecting fake news without affecting the performance in the real news class.
2. **RQ2: Which GAN architectures are more suitable for the generation of tabular data?** From the experiments in the 5.3 and 5.4 sections, we saw that the TabFairGAN architecture gave the best results in terms of intrinsic data quality, but this was not reflected in the best results for data augmentation. The CTGAN architectures provided better results in data augmentation than the TabFairGAN architecture.
3. **RQ3: How does the intrinsic quality of synthetic data influence the results of classification in a data augmented dataset?** While searching for the answer to **RQ2**, we came across the answer for **RQ3**. From the experiments we conducted, we observed that the higher the quality of the synthetic data, the higher the performance gain it could bring in the minority class. However, this could be at the cost of performance in the majority class, as the generated and evaluated data concerned only the minority class.

6.2 Contributions

In the course of answering these research questions, this study offers a number of contributions, which are listed below.

- We conducted a comprehensive review of synthetic data generation methods, GANs and synthetic data quality evaluation. To our knowledge, no other study in the relevant literature has explicitly combined these topics. In particular, we focused on GANs for tabular data generation, a very understudied topic compared to image generation. This work fills this gap and provides useful material for new researchers in this area.

- We have shown that data augmentation can indeed be used to enhance the performance of ML models in a public and well-known news dataset (LIAR-PLUS). Furthermore, we have shown which GAN architectures are better suited for generating tabular data, both in terms of the utility framework (section 5.3) and data augmentation (section 5.4).

- We have assessed the data quality of the datasets in a general way, making the connection between data quality and data augmentation performance with the use of a knowledgeably modified and comprehensive utility framework.

- We published a paper [83] at the WorldCist'22 conference, which served as the motivation for this work.

- We published a paper [31] in the MDPI Mathematics Journal, a Q1 (WoS) journal, showing the state-of-the-art review conducted in this work.

6.3 Limitations

One of the limitations of this work concerns the simplification of using GANs for tabular data generation while Fake News are, typically, represented in text format – tweets, Facebook posts, news websites, etc. As such, when transforming text into tabular data some information is lost in the process. Dealing directly with news in its raw form might enable us to better address the problem, but it would also add a fair share of complexity.

In line with the previous limitation, in this work we have used a utility framework to evaluate synthetic data which was very encompassing, but lacked the ability of assessing text data, even if we had generated it. As such, another limitation is the inability to evaluate text data with the methods we have used.

Finally, given the three broad topics we have combined in this work – synthetic data generation methods, GANs and evaluation of synthetic samples – some details have not been included. Indeed, the literature is so vast that it would be impossible to summarize everything in a single work.

6.4 Future Work

Taking into account the previous section, there are multiple ways for extending this work. Firstly, hyper-parameter tuning could be applied using more hyperparameters, and the number of training epochs could take longer. This would allow for better trained GANs and, probably, better quality synthetic data. Additionally, other generative models could be used and compared with GANs, such as Autoencoders (AEs), flow-based models, or diffusion models.

Another way we could improve this work is by considering news datasets in raw form (text format) and address them as they are. When news are represented as tabular data, some information is lost in the process. By investigating about GANs for text generation, perhaps we can boost the results.

As seen in section 5.4, the GAN that showed better results regarding the multiple utility metrics used (see the radar chart in figure 5.6 for a quick refresher) was not the one that provided the best results in the data augmentation task. This was quite interesting and we believe this should be further explored. Not only to better understand the relation between the intrinsic data quality and the data augmentation classification performance but also to assess if there are more informative measures regarding the data augmentation.

In section 5.3 we have noted that the features *capital-gain* and *capital-loss* were not included in this synthetic data evaluation analysis because they were poorly generated by all the GAN models (see figure 5.1). This, however, may be addressed in future work to understand what caused this to happen and, thus, to explore the relationship between features with a certain distribution and the impact they have on the performance of GANs.

Another aspect we want to improve is the use of more test sets in order to decrease the bias that may have been (inadvertently) introduced. As mentioned in section 5.4, both the Adult and the LIAR-PLUS datasets were split into train and test sets. However, despite this random split, some bias may have been introduced. As such, the replication of the experiments with more train-test splits could be important to address the matter.

In what concerns the experiments in section 5.4, some further analysis can prove fruitful to shed some light on what causes abrupt gains/losses in ML performance when the number of samples increases. More specifically, we want to fully comprehend what caused the performance gains (in the minority class) in the Adult dataset when the proportion of minority class samples generated by the TabFairGAN increased from 0.9 to 1 in the accuracy and F1 of the logistic regression, as well as the abrupt losses in precision and recall (see figure 5.7). Moreover, in the LIAR-PLUS dataset, the wild oscillations in the performance of the minority class of the CTGAN 4 generated samples for the precision of the decision tree (see figure 5.11) are also worth of further exploration.

Furthermore, it became clear from our experience during this study that GANs is under-researched for generating tabular data compared to GANs for synthesizing image data. Therefore, we could extend this work by creating our own GAN for generating synthetic tabular data. In addition, we would be interested in creating a package that other researchers can easily use without them having to implement it themselves. Such a package could provide researchers with several built-in tabular GAN architectures, removing the need for the user to build them from scratch. Moreover, it could have a module for synthetic data evaluation, which would allow users to get immediate feedback as soon as their GAN are trained. This way, users could rapidly try out several GAN architectures and see which best fit their use case. This would be very useful since the existing packages still have a lot of room for improvement.

We believe that the approach proposed in this paper has several applications. Since the architectures we use are not limited to news datasets, they can be applied to other domains where tabular data is used. In addition, they could help detect fake news that would not be captured if the ML models were trained only on an unexpanded dataset. Clearly, this can have a positive impact and help mitigate a serious problem that affects us all: Fake News.

Bibliography

[1] Annisa Aditsania, Aldo Lionel Saonard, et al. Handling imbalanced data in churn prediction using adasyn and backpropagation algorithm. In *2017 3rd international conference on science in information technology (ICSITech)*, pages 533–536. IEEE, 2017.

[2] Ahmed M. Alaa, Boris van Breugel, Evgeny Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models, 2021.

[3] Yasir Alanazi, Nobuo Sato, Pawel Ambrozewicz, Astrid N Hiller Blin, Wally Melnitchouk, Marco Battaglieri, Tianbo Liu, and Yaohang Li. A survey of machine learning-based physics event generation. *arXiv preprint arXiv:2106.00643*, 2021.

[4] Adamu Ali-Gombe and Eyad Elyan. Mfc-gan: class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing*, 361:212–221, 2019.

[5] Adamu Ali-Gombe, Eyad Elyan, Yann Savoye, and Chrisina Jayne. Few-shot classifier gan. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[6] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[7] Gerard Andrews. What is synthetic data? https://blogs.nvidia.com/blog/2021/06/08/what-is-synthetic-data/, Jun 2021.

[8] Francis J Anscombe. Graphs in statistical analysis. *The american statistician*, 27(1):17–21, 1973.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[10] Samuel Assefa. Generating synthetic data in finance: opportunities, challenges and pitfalls. *Challenges and Pitfalls (June 23, 2020)*, 2020.

[11] Joris Baan. A comprehensive introduction to bayesian deep learning. https://jorisbaan.nl/2021/03/02/introduction-to-bayesian-deep-learning, Mar 2021.

[12] Ruud Barth, JMM IJsselmuiden, Jochen Hemming, and Eldert J van Henten. Optimising realism of synthetic agricultural images using cycle generative adversarial networks. In *Proceedings of the IEEE IROS workshop on Agricultural Robotics*, pages 18–22, 2017.

[13] Rukshan Batuwita and Vasile Palade. Efficient resampling methods for training support vector machines with imbalanced datasets. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[14] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[15] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 475–482. Springer, 2009.

[16] Weiwei Cai and Zhanguo Wei. Piigan: generative adversarial networks for pluralistic image inpainting. *IEEE Access*, 8:48451–48463, 2020.

[17] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.

[18] Jianhui Chen and James J Little. Sports camera calibration via synthetic data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[19] Shuanglian Chen. Research on extreme financial risk early warning based on odr-adasyn-svm. In *2017 International Conference on Humanities Science, Management and Education Technology (HSMET 2017)*, pages 1132–1137. Atlantis Press, 2017.

[20] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2180–2188, 2016.

[21] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[22] Chanachok Chokwitthaya, Yimin Zhu, Supratik Mukhopadhyay, and Amirhosein Jafari. Applying the gaussian mixture model to generate large synthetic data from a small data set. In *Construction Research Congress 2020: Computer Applications*, pages 1251–1260. American Society of Civil Engineers Reston, VA, 2020.

[23] CKCN Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[24] João Coutinho-Almeida, Pedro Pereira Rodrigues, and Ricardo João Cruz-Correia. Gans for tabular healthcare data generation: A review on utility and privacy. In Carlos Soares and Luis Torgo, editors, *Discovery Science*, pages 282–291, Cham, 2021. Springer International Publishing. ISBN: 978-3-030-88942-5.

[25] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

[26] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465: 1–20, 2018.

[27] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003.

[28] Khaled Emam, Lucy Mosquera, and Richard Hoptroff. Chapter 1: Introducing synthetic data generation. In *Practical Synthetic Data Generation: Balancing Privacy and the broad availability of data*, page 1–22. O'Reilly Media, Inc., 2020.

[29] Khaled Emam, Lucy Mosquera, and Richard Hoptroff. Chapter 4: Evaluating synthetic data utility. In *Practical Synthetic Data Generation: Balancing Privacy and the broad availability of data*, page 69–94. O'Reilly Media, Inc., 2020.

[30] Fatemeh Fahimi, Zhuo Zhang, Wooi Boon Goh, Kai Keng Ang, and Cuntai Guan. Towards eeg generation using gans for bci applications. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 1–4. IEEE, 2019.

[31] Alvaro Figueira and Bruno Vaz. Survey on synthetic data generation, evaluation methods and gans. *Mathematics*, 10(15), 2022. ISSN: 2227-7390. doi:10.3390/math10152733.

[32] David Foster. Chapter 3: Variational autoencoders. In *Generative deep learning: Teaching machines to paint, write, compose, and play*, page 61–96. O'Reilly, 2019.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Chapter 20: Deep generative models. In *Deep Learning*, pages 654–720. MIT Press, 2016.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Chapter 14: Autoencoders. In *Deep Learning*, pages 502–525. MIT Press, 2016.

[36] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[37] Max Hänska and Stefan Bauchowitz. *Tweeting for Brexit: how social media influenced the referendum.* abramis academic publishing, 2017.

[38] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[39] E. Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 1909(136):210–271, 1909. doi:doi:10.1515/crll.1909.136.210 [visited 2022-09-12].

[40] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[41] Zubayer Islam, Mohamed Abdel-Aty, Qing Cai, and Jinghui Yuan. Crash data augmentation using variational autoencoder. *Accident Analysis & Prevention*, 151:105950, 2021.

[42] Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *ACM Sigkdd Explorations Newsletter*, 6(1):40–49, 2004.

[43] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[44] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[45] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[46] Lan Lan, Lei You, Zeyang Zhang, Zhiwei Fan, Weiling Zhao, Nianyin Zeng, Yidong Chen, and Xiaobo Zhou. Generative adversarial networks and its applications in biomedical informatics. *Frontiers in Public Health*, 8, 2020. ISSN: 2296-2565. doi:10.3389/fpubh.2020.00164.

[47] Scikit Learn. Gaussian mixture models. https://scikit-learn.org/stable/modules/mixture.html, 2022.

[48] Taejun Lee, Minju Kim, and Sung-Phil Kim. Data augmentation effects using borderline-smote on classification of a p300-based bci. In *2020 8th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–4. IEEE, 2020.

[49] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29:469–477, 2016.

[50] Chao Lu, Shaofu Lin, Xiliang Liu, and Hui Shi. Telecom fraud identification based on adasyn and random forest. In *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, pages 447–452. IEEE, 2020.

[51] Lara Lusa et al. Evaluation of smote for high-dimensional class-imbalanced microarray data. In *2012 11th international conference on machine learning and applications*, volume 2, pages 89–94. IEEE, 2012.

[52] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

[53] Matthew Mayo. Yann lecun quora session overview. https://www.kdnuggets.com/2016/08/yann-lecun-quora-session.html, Aug 2016.

[54] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[55] Robert S Mueller et al. *The Mueller Report*. e-artnow, 2019.

[56] Sergey I Nikolenko et al. Synthetic data for deep learning. *arXiv preprint arXiv:1909.11512*, 3, 2019.

[57] Murphy Yuezhen Niu, Alexander Zlokapa, Michael Broughton, Sergio Boixo, Masoud Mohseni, Vadim Smelyanskyi, and Hartmut Neven. Entangling quantum generative adversarial networks. *Physical Review Letters*, 128(22):220505, 2022.

[58] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[59] Ofcom. News consumption in the uk: Research report - ofcom, Dec 2015.

[60] Femi Olan, Uchitha Jayawickrama, Emmanuel Ogiemwonyi Arakpogun, Jana Suklan, and Shaofeng Liu. Fake news on social media: the impact on society. *Information Systems Frontiers*, pages 1–16, 2022.

[61] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*, 2018.

[62] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[63] Mansi Patel, Xuyu Wang, and Shiwen Mao. Data augmentation with conditional gan for automatic modulation classification. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, pages 31–36, 2020.

[64] Tim Prangemeier, Christoph Reich, Christian Wildner, and Heinz Koeppl. Multi-stylegan: Towards image-based simulation of time-lapse live-cell microscopy. *arXiv preprint arXiv:2106.08285*, 2021.

[65] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[66] Amirarsalan Rajabi and Ozlem Ozmen Garibay. Tabfairgan: Fair tabular data generation with generative adversarial networks. *arXiv preprint arXiv:2109.00666*, 2021.

[67] Julio C. S. Reis, André Correia, Fabrício Murai, Adriano Veloso, and Fabrício Benevenuto. Supervised learning for fake news detection. *IEEE Intelligent Systems*, 34(2):76–81, 2019. doi:10.1109/MIS.2019.2899143.

[68] D Riafio et al. Using gabriel graphs in borderline-smote to deal with severe two-class imbalance problems on neural networks. In *Artificial Intelligence Research and Development: Proceedings of the 15th International Conference of the Catalan Association for Artificial Intelligence*, volume 248, page 29. IOS Press, 2012.

[69] Stuart Jonathan Russell, Peter Norvig, and Ming-Wei Chang. Chapter 13: Probabilistic reasoning. In *Artificial Intelligence: A modern approach*, page 430–478. Pearson, 2022.

[70] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[71] Sobhan Sarkar, Anima Pramanik, J Maiti, and Genserik Reniers. Predicting and analyzing injury severity: A machine learning-based approach using class-imbalanced proactive and reactive data. *Safety science*, 125:104616, 2020.

[72] Karen-Beth G Scholthof. The disease triangle: pathogens, the environment and society. *Nature Reviews Microbiology*, 5(2):152–156, 2007.

[73] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2018.

[74] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1): 22–36, 2017.

[75] Bhargav Siddani, S Balachandar, William C Moore, Yunchao Yang, and Ruogu Fang. Machine learning for physics-informed generation of dispersed multiphase flow using generative adversarial networks. *Theoretical and Computational Fluid Dynamics*, 35(6): 807–830, 2021.

[76] Wacharasak Siriseriwan and Krung Sinapiromsaran. The effective redistribution for imbalance dataset: relocating safe-level smote with minority outcast handling. *Chiang Mai Journal of Science*, 43(1):234–246, 2016.

[77] Devin Soni. Introduction to bayesian networks. https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e, Jul 2019.

[78] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *Advances in neural information processing systems*, 30, 2017.

[79] Jie Sun, Jie Lang, Hamido Fujita, and Hui Li. Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. *Information Sciences*, 425:76–91, 2018.

[80] Jie Sun, Hui Li, Hamido Fujita, Binbin Fu, and Wenguo Ai. Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting. *Information Fusion*, 54:128–144, 2020.

[81] Vadim Sushko, Jurgen Gall, and Anna Khoreva. One-shot gan: Learning to generate samples from single images and videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2596–2600, 2021.

[82] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[83] Bruno Vaz, Vítor Bernardes, and Álvaro Figueira. On creation of synthetic samples from gans for fake news identification algorithms. In *World Conference on Information Systems and Technologies*, pages 316–326. Springer, 2022.

[84] Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 647–653, 2017.

[85] Zhiqiang Wan, Yazhou Zhang, and Haibo He. Variational autoencoder based synthetic data generation for imbalanced learning. In *2017 IEEE symposium series on computational intelligence (SSCI)*, pages 1–7. IEEE, 2017.

[86] William Yang Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[87] William Weir. *History's Greatest Lies: The Startling Truths Behind World Events Our History Books Got Wrong.* Fair winds press, 2009.

[88] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

[89] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *arXiv preprint arXiv:1907.00503*, 2019.

[90] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked

generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[91] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[92] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.

[93] Weiyu Zhang, Yiyang Ma, Di Zhu, Lei Dong, and Yu Liu. Metrogan: Simulating urban morphology with generative adversarial network. *arXiv preprint arXiv:2207.02590*, 2022.

[94] Xi Zhang, Yanwei Fu, Andi Zang, Leonid Sigal, and Gady Agam. Learning classifiers from synthetic data using a multichannel autoencoder. *arXiv preprint arXiv:1503.03163*, 2015.

[95] Xinyi Zhou, Atishay Jain, Vir V Phoha, and Reza Zafarani. Fake news early detection: A theory-driven model. *Digital Threats: Research and Practice*, 1(2):1–25, 2020.

[96] Yuan Zhou, Fang Dong, Yufei Liu, Zhaofu Li, JunFei Du, and Li Zhang. Forecasting emerging technologies using data augmentation and deep learning. *Scientometrics*, 123(1): 1–29, 2020.

[97] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.